ANALYSES AVANCÉES DE LA MÉTHODE HYBRIDE GMRES/LS-ARNOLDI ASYNCHRONE PARALLÈLE ET DISTRIBUÉE POUR LES GRILLES DE CALCUL ET LES SUPERCALCULATEURS

Haiwu He

THÈSE

Présentée à

L'UNIVERSITÉ DES SCIENCES ET TECHNOLOGIES DE LILLE

Pour obtenir le titre de

DOCTEUR EN INSTRUMENTATIONS ET ANALYSES AVANCÉES

par

Haiwu HE

ANALYSES AVANCÉES DE LA MÉTHODE HYBRIDE GMRES/LS-ARNOLDI ASYNCHRONE PARALLÈLE ET DISTRIBUÉE POUR LES GRILLES DE CALCUL ET LES SUPERCALCULATEURS

Thèse soutenue le 08 juillet 2005, devant la Commission d'Examen :

Président:	Jean-Marc GEIB,	Université de Lille I
Rapporteurs:	Pierre MANNEBACK,	Faculté polytechnique de Mons
	Nahid EMAD	Laboratoire PRISM
Examinateurs:	Serge PETITON,	Université de Lille I
	Zhi-Jian WANG,	Université de Hohai
	Mitsuhisa SATO,	Université de Tsukuba
	Guy BERGERE,	LIFL

A ma famille HE et mon pays la Chine,

Remerciements

En tout premier lieu, je remercie M. Serge PETITON, Professeur à l'Université des Sciences et Technologies de Lille qui a dirigé cette thèse durant trois années, dans la continuité de mes travaux de D.E.A. J'ai pu profiter de sa compétence exceptionnelle. Il a toujours été disponible quand j'ai rencontré des difficultés dans ma recherche. Comme un thésard étranger, au début de ma thèse, il y avait une grande difficulté de langue. Grâce à sa grande patience, ses encouragements et sa bienveillance personnelle, j'ai pu surmonter l'obstacle de la langue française et aller plus loin dans la recherche. Je le remercie vivement.

Je remercie particulièrement Dr. Guy BERGERE du Laboratoire d'Informatique Fondamentale de Lille. Nous avons fait le point chaque semaine, à discuter de l'avancement de notre recherche, également des obstacles et problèmes. Il m'a donné beaucoup de conseils précieux pour mon travail. Il a apporté également de nombreuses suggestions et corrections au manuscrit. J'ai obtenu beaucoup d'expérience et d'inspiration de son travail.

Je remercie les rapporteurs de cette thèse M. Pierre MANNEBACK, Professeur à la Faculté Polytechnique de Mons, ainsi que Mme Nahid EMAD, Maître de Conférence (HDR) à l'Université de Versailles, d'avoir accepté de rapporter ma thèse et pour les échanges constructifs en ayant résulté et l'intérêt qu'ils ont porté à mon travail.

Je tiens à remercier M. Jean-Marc GEIB, Professeur à l'Université de Sciences et Technologies de Lille et le directeur du Laboratoire d'Informatique Fondamentale de Lille, pour son soutien en continu pendant mes quatre années d'études au LIFL et pour avoir accepté de présider le jury de cette thèse.

Je remercie M. Zhijian WANG, Professeur à l'Université de HOHAI à Nanjing en Chine pour avoir accepté d'être membre du jury et pour être venu de la Chine lointaine pour m'honorer de sa présence lors de ma soutenance.

Je remercie Mitsuhisa SATO, Professeur à l'Université de TSUKUBA au Japon pour avoir accepté d'être membre du jury et pour être venu du Japon lointain pour m'honorer de sa présence lors de ma soutenance.

Je tiens particulièrement à remercier M. Aouad Lamine Mohamed, de l'équipe MAP du Laboratoire d'Informatique Fondamentale de Lille, pour sa collaboration dans l'utilisation du système XtremWeb et pour sa gentillesse et beaucoup d'aides pendant les trois ans de ma thèse.

Je remercie Mme Nicole FLINOIS et Mme Bérengère DOBY-DASSONVILLE pour leurs accueils chaleureux au Laboratoire d'Informatique Fondamentale de Lille et pour leurs gentillesses et beaucoup d'aides pendant les quatre ans de mes études en France.

Je remercie également les autres membres de notre équipe MAP, M. Laurent CHOY, M. Toussaint GUGLIELMI pour leurs supports généreux.

Je remercie aussi M. Richard OLEJNIK, M. Bruno BOURISER et les autres collègues du Laboratoire d'Informatique Fondamentale de Lille pour leur soutien généreux.

Je remercie également mes camarades chinois qui viennent de l'Université de HOHAI , Dr XIE Shouyi, Mme SU Man, M. HU Hexuan, M. ZHU Qizhi, M. SHI Xiaodong, M.WANG Wei, M. CHEN Da, Mme JIA Yun, et Mme XU Jian, pour leurs soutiens généreux.

Résumé

De nombreux problèmes scientifiques et industriels ont besoin de la résolution de systèmes linéaires non symétriques à grande échelle, qui sont décrits par des matrices creuses de très grande taille. On utilise fréquemment dans ce cas des méthodes numériques itératives et on fait appel au parallélisme pour une résolution rapide et efficace. L'algorithme GMRES(m) est une méthode itérative qui donne de bons résultats dans la plupart des cas. Mais on observe une limitation à sa parallélisation en raison des nombreuses communications produites. Dans quelques cas, la convergence est atteinte très lentement, voire jamais. Nous présentons dans cette thèse une méthode hybride GMRES(m)/LS-Arnoldi qui accélère la convergence grâce à la connaissance des valeurs propres calculées parallèlement par la méthode d'Arnoldi pour les cas réels, avec son implantation sur des supercalculateurs. Une extension aux cas complexes est également étudiée.

La dernière tendance du calcul global, le calcul de grille, propose l'exploitation massive des ressources vacantes des réseaux locaux ainsi que sur Internet. Son avantage peut être énorme pour l'exécution d'applications parallèles. L'environnement XtremWeb est un système de grille léger, tolérant aux défaillances et sécurisé pour l'exécution d'applications parallèles. Il est un environnement de calcul haute-performance, une plate-forme de grille logicielle d'expérimentation pour des institutions académiques ou industrielles. Nous présentons dans cette thèse les implantations de la méthode GMRES(m) sur ce système de grille XtremWeb ainsi que sur un environnement distribué de calcul LAM-MPI.

Nous avons fait de multiples tests sur grille et supercalculateur. Des performances que nous avons obtenues, nous constatons les avantages et les inconvénients de ces plates-formes de calcul différentes.

Mots clefs : le calcul de grille, matrices creuses, méthodes hybrides, parallélisme asynchrone, GMRES, XtremWeb, LAM-MPI, pair à pair

Abstract

Many scientific and industrial problems need the resolution of nonsymmetric linear systems of large scale, which are described by sparse matrices of very large size. We frequently use the iterative numerical methods and benefit from parallelism for a fast and effective resolution. The GMRES(*m*) algorithm is an iterative method which gives good results in most cases. Nevertheless we observe the limitation of its parallelization because of much provoked communications, in some case convergence is reached very slowly even never. We present in this thesis a hybrid method GMRES(*m*)/LS-Arnoldi which accelerates the convergence thanks to the knowledge of the eigenvalues calculated in parallel by the method of Arnoldi for the real cases with its implementation on the supercomputers. Furthermore we study an extension of complex cases.

The latest tendency of global computing, the GRID computing proposes the massive exploitation of the vacant resources on the local area networks and on world wide Internet for the execution of parallel applications. The XtremWeb environment is a secured, light GRID system, with the failures tolerance mechanism for the execution of parallel applications. It is a high-performance computing environment, a software GRID platform of experimentation for academic or industrial organisation. We present in this thesis the implementation of the GMRES(*m*) method on this GRID system XtremWeb as well as a distributed computing environment LAM-MPI.

We made numerous tests on GRID and supercomputer. From performances which we obtained, we note the advantages and the disadvantages for these different computing platforms.

Key words: GRID computing, sparse matrices, hybrid method, asynchronous parallelism, GMRES(*m*), XtremWeb, LAM-MPI, peer to peer.

Chapitre 1 Introduction

La résolution de systèmes linéaires à grande échelle est un important problème scientifique ou industriel. Ces systèmes linéaires sont souvent révélés sous la forme de matrices non symétriques creuses et de très grande taille. De multiples phénomènes physiques sont modélisés mathématiquement à l'aide d'équations aux dérivées partielles (EDP) qui, après discrétisation, aboutissent souvent à résoudre un système linéaire représenté par l'équation algébrique

$$Ax = b$$

La résolution de tels systèmes est l'objet de nombreuses recherches aussi bien à propos des méthodes numériques utilisées qu'au sujet de leur réalisation informatique. La taille des problèmes concernés et leur coût en temps de calcul incite à optimiser les méthodes employées et impose l'usage du parallélisme.

Le travail présenté ici à ce propos a été réalisé au sein de l'équipe MAP du Laboratoire d'Informatique Fondamentale de Lille (LIFL).

Cette thèse développe dans un premier temps la mise au point de la méthode hybride GMRES(*m*)/LS-Arnoldi élaborée à partir de la méthode traditionnelle GMRES(*m*) qui a une limitation évidente à cause du grand nombre des communications engendrées (voir la figure 3.2.2). Les méthodes numériques utilisées décrites dans la section 3.2 sont des méthodes itératives. Vu sous l'aspect numérique, la caractéristique de l'approche du problème demeure dans l'hybridation des méthodes « GMRES » pour la résolution du système, « Arnoldi » pour le calcul des valeurs propres et « Least Squares » pour la prise en compte de celles-ci dans le calcul d'un nouvel itéré permettant l'accélération de la convergence.

L'aspect informatique du problème est aussi peu commun, parce qu'il profite à la fois du parallélisme, et de l'asynchronisme. En fait, comme cela sera explicité dans la section 3.2.1, le degré de parallélisation permis par les méthodes numériques telles que GMRES et Arnoldi est restreint par le grand nombre des communications entraînées. Pour améliorer les performances, il faut trouver d'autres moyens que la simple augmentation du nombre des processeurs participant au calcul.

L'hybridation de certaines méthodes numériques est un cas idéal : En assignant les calculs de chaque méthode à une machine parallèle différente, ou à une partition différente de processeurs d'une même machine, on profite mieux du parallélisme potentiel.

Deux implantations de la méthode GMRES mentionnée ont été mises en place :

Une version asynchrone, répartie entre les processeurs d'un supercalculateur IBM SP3, est décrite au Chapitre 4 et au Chapitre 5. La réalisation et l'intérêt de la méthode hybride GMRES(*m*)/LS-Arnoldi dans le cas réel sont détaillés au Chapitre 4, une extension expérimentale dans le cas complexe est montrée au Chapitre 5.

La croissance exponentielle du nombre de PCs connectés sur Internet, la popularisation des technologies de réseau haut débit au foyer et la disponibilité dans ces réseaux de nombreuses ressources informatiques largement inexploitées offrent l'opportunité de mettre en place une grille de calcul qui globalise l'accès aux ressources et aux données. Pour profiter de cette situation, nous adaptons notre algorithme GMRES(*m*) qui fonctionne bien dans l'environnement de supercalculateur à une grille légère XtremWeb.

Une version distribuée, hétérogène, utilisant un système distribué pour le calcul scientifique, le système de grille légère XtremWeb, est présentée au Chapitre 6. Elle est comparée avec la version précédente adaptée à un environnement distribué de calcul de LAM-MPI.

Dans les deux implantations, les résultats obtenus présentent parallèlement l'intérêt de la méthode numérique et des choix à l'égard de sa mise en œuvre informatique. Deux sujets importants, liés à tout calcul parallèle et distribué sur les matrices creuses, ont été peu discutés dans cette thèse car ils auront besoin de développements très importants :

Le cas des matrices logées partiellement en mémoire. Un découpage par bloc est alors nécessaire ce qui implique un partitionnement de la matrice creuse. L'optimisation de ce découpage est un problème NP-complet [57].

L'optimisation du placement des matrices. C'est un problème particulièrement important pour obtenir une parallélisation optimale des communications avec une architecture massivement parallèle ou distribuée [55],[56].

Cette thèse est divisée en chapitres comme suit :

Chapitre 2 Nous faisons l'analyse numérique parallèle et distribuée de notre méthode traditionnelle et hybride, nous présentons les deux types de machines parallèles avec leurs architectures différentes. Après une introduction brève des deux supercalculateurs que nous avons utilisés pour nos tests , nous décrivons les conceptions de quatre systèmes de calcul, le GRID, le P2P, XtremWeb ainsi qu'un environnement parallèle de calcul sous MPI : LAM-MPI.

Chapitre 3 Nous détaillons les formats des matrices creuses que nous avons employés pour profiter du parallélisme et éviter de consommer trop de mémoire. Ensuite, nous présentons les développements des algorithmes employés, depuis celui de la méthode traditionnelle GMRES jusqu'à celui beaucoup plus compliqué, de la méthode hybride GMRES(m)/LS-Arnoldi.

Chapitre 4 Nous précisons notre implantation de l'algorithme parallèle asynchrone sur les machines à plusieurs nœuds à mémoire partagée. Nous en décrivons une organisation générale, ainsi que le mécanisme d'accumulation des valeurs propres. Nous présentons l'asynchronisme que nous rencontrons et quelques précautions nécessaires. Enfin, nous montrons les résultats obtenus, puis nous analysons les paramètres importants ayant une grande influence sur cette méthode.

Chapitre 5 Dans le chapitre précédent, nous avons utilisé la méthode hybride GMRES(*m*)/LS-Arnoldi qui accélère beaucoup la convergence des matrices pendant les calculs dans le cas de variables réelles. Dans ce chapitre, nous essayons d'appliquer la même méthode hybride après avoir effectué les transformations permettant de passer du cas réel au cas complexe. Nous présentons nos difficultés théoriques et tentons de trouver une solution.

Chapitre 6 Dans le contexte moderne du calcul de grille, nous essayons d'utiliser l'algorithme GMRES(*m*) pour profiter des ressources inexploitées sur Internet. Nous présentons le système léger de grille XtremWeb et notre implantation sur cette plate-forme de calcul. Nous avons effectué également ces tests dans un environnement distribué de calcul de LAM-MPI, dont nous comparons les résultats obtenus. En analysant les performances de ces tests, nous remarquons les avantages et les inconvénients de notre réalisation en plate-forme distribuée d'Intranet ou d'Internet.

Chapitre 7 Ce chapitre est la conclusion de cette thèse et décrit les perspectives de recherches futures dans ce domaine.

Chapitre 2 Analyse numérique parallèle et distribuée

Pour présenter le plus clairement possible la parallélisation effectuée, nous rappelons certaines notions de base et nous présentons la machine ciblée. On donnera une brève description du parallélisme et des deux classes de machine parallèles SIMD et MIMD, en particulier des machines que nous utiliserons dans l'implantation des algorithmes parallèles : IBM SP3, IBM SP4. Nous présenterons aussi quelques éléments essentiels pour paralléliser des algorithmes sur des machines parallèles. Nous montrerons également la conception du GRID, les descriptions du P2P, un système GRID léger XtremWeb et un système distribué LAM-MPI (Local Area Multi-computer MPI).

2.1 Parallélisme en analyse numérique

Pour beaucoup de domaines scientifiques et industriels, la demande de méthodes d'analyse numérique et d'outils est de plus en plus importante, surtout pour les résolutions des grands systèmes linéaires et des problèmes scientifiques aux valeurs propres. Fréquemment ces problèmes dérivent de discrétisation d'équations aux dérivées partielles (EDP) et sollicitent le traitement de matrices de très grande taille.

Le nombre important de données cause des problèmes de stockage et de temps de calcul qui sont excessifs sur les machines séquentielles (selon Von Neumann), d'où l'exigence d'utiliser de machines parallèles ou des systèmes distribués pour surmonter les limitations. Les grandes machines (les supercalculateurs) font souvent collaborer jusqu'à une centaine de processeurs et ont une grande capacité de mémoire. Et les systèmes distribués font coopérer jusqu'à une dizaine de milliers de PCs avec des systèmes d'exploitation différents. Souvent, les systèmes distribués à grande échelle sont une version économique de supercalculateur. La minimisation du temps de calcul parallèle découle de la

réalisation de plusieurs opérations simultanément. La manière dont les processeurs sont connectés, leurs composants et leur mode de contrôle conduit à l'obtention de plusieurs genres d'architectures.

2.2 Machine Parallèle

Selon la façon d'accès à la mémoire, les machines parallèles sont réparties en deux grandes sortes :

Les machines à mémoire partagée: Littéralement, tous les processeurs des ces machines accèdent aux données dans une même mémoire. Dans ce cas, l'utilisateur n'a pas besoin de savoir le lieu des données ni de manager les communications parmi les processeurs. Grâce à cette architecture, la conception de programme sur ces machines est plus simple, mais le problème de l'accès à la mémoire restreint le nombre de processeurs utilisé.

Les machines à mémoire distribuée: Chaque processeur de ces machines a sa propre mémoire et a ses propres données. Sachant que ces processeurs travaillent sur la même application dont les données sont réparties, il est évidemment nécessaire qu'ils se communiquent pour échanger leurs données au cours de l'exécution de l'application.

Comparé à la première classe de machines parallèles, la seconde classe peut utiliser des tailles mémoires très grandes. Cependant, l'efficacité des programmes sur les machines à mémoire distribuée peut seulement être obtenue par les placements judicieux des données. Afin d'exploiter la localité des données pour minimiser le coût des communications parmi les processeurs et les mémoires. Dans ma thèse nous n'utilisons que des machines parallèles à mémoire distribuée qui sont plus en plus populaires.

Se basant sur la façon dont sont traités le flot d'instructions et le flot de données, Flynn [2] classe et compare les machines parallèles. Selon cette terminologie, il y a deux classes de machines parallèles : SIMD et MIMD [3], [4]. Avant de décrire

ces deux classes de machines parallèles en détail, on note qu'un processeur est composé de trois éléments : l'unité de contrôle (décode les instructions), l'unité de traitement (exécute les opérations) et la mémoire (stocke les données).

2.2.1 Machine SIMD

SIMD signifie Simple Instruction Multiple Data ; c'est un ensemble de processeurs surveillés par la même unité de contrôle, qui exécutent la même instruction en même temps, chacun sur ses propres données possédés dans sa propre mémoire locale. Un ordinateur appelé frontal (FE, Front End) auquel sont reliés tous les processeurs à travers l'unité de contrôle permet l'accès à la machine parallèle. Généralement les processeurs de ces machines sont nombreux, mais ils ne sont pas très puissants. Ce genre de machines conduit souvent à une programmation basée sur le parallélisme de données. Sur ces machines, on procède à la parallélisation des données des applications dont le parallélisme potentiel est maximal. L'architecture des machines est représentée dans la FIG 2.1

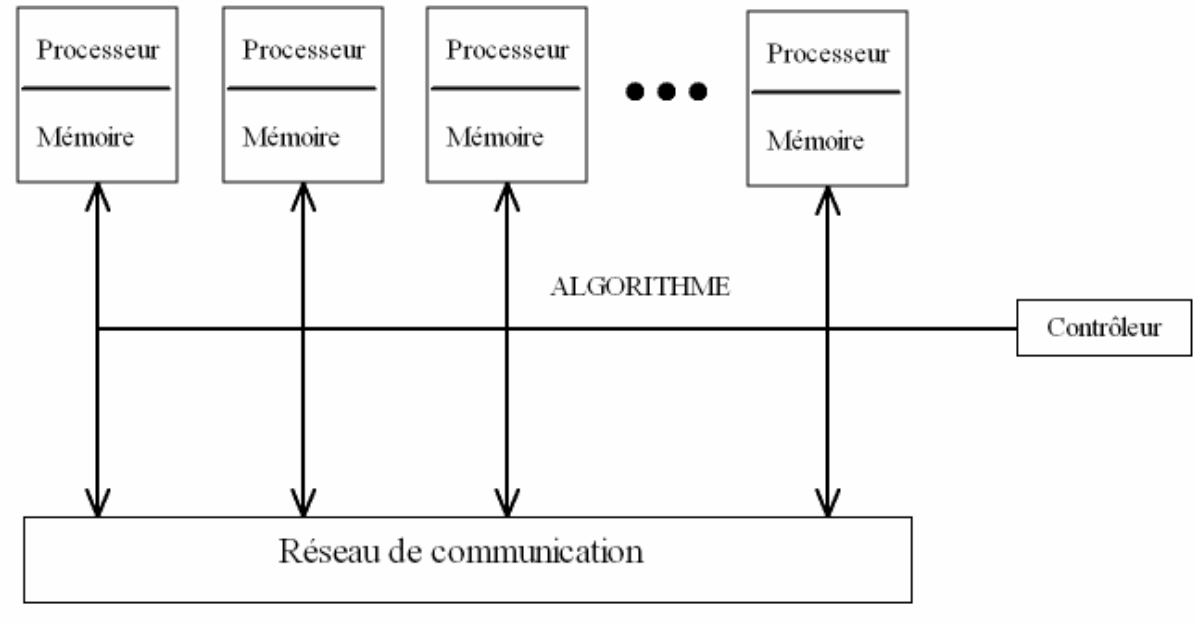


FIG 2.1 -L'architecture des machines SIMD

2.2.2 Machines MIMD et SPMD

MIMD signifie **M**ultiple **I**nstruction **M**ultiple **D**ata ; c'est un ensemble de processeurs dont chacun d'entre eux a sa propre mémoire contenant des données locales sur lesquelles s'exécute son propre programme. Chacun processeur peut donc travailler indépendamment des autres processeurs. Ces machines n'ont pas de frontaux, mais chacun de ces processeurs peut prendre cette fonction. Ordinairement les processeurs de ces machines ne sont pas très nombreux, mais ils sont très puissants. Ce genre de machines ont deux modèles de programmations : parallélisme de données et parallélisme de tâches. Les machines **MIMD** fonctionnent en mode **SPMD** (**S**ingle **P**rogramme **M**ultiple **D**ata) lorsqu'un même programme s'exécute sur tous les processeurs. L'architecture des machines **MIMD** est représentée dans FIG 2.2.

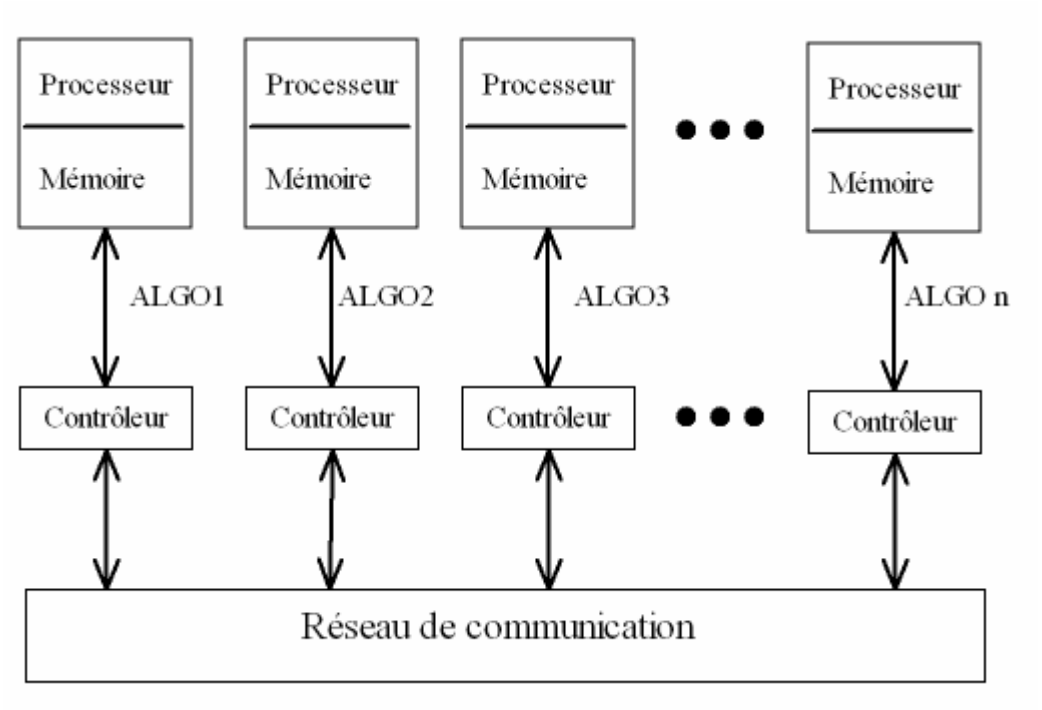


FIG 2.2-L'architecture des machines MIMD

Les échanges de données parmi les différents processeurs sur ces machines sont réalisés à l'aide d'une bibliothèque de communications, comme par exemple la bibliothèque PVM [5](Parallel Virtual Machine) ou la bibliothèque MPI (Message Passing Interface) [6].

2.3 Description de IBM SP3 et SP4

Le supercalculateur à architecture parallèle IBM RS6000/SP3 que nous utilisons est installé au Centre de Ressources Informatiques de l'Université des Sciences et Technologies de Lille. Il est destiné aux chercheurs de toutes disciplines confrontés à des calculs intensifs et donc demandeurs de puissance de calcul et de grandes capacités de mémoire [7].

4 nœuds hauts, chacun comprenant :

- 16 processeurs Power3 NH2 à 375Mhz
- 16 Go RAM
- 2 disques de 18.2 Go
- Les nœuds sont interconnectés par un réseau haut débit colony
- 12 disques externes SSA de 36.4 Go accessibles des 4 nœuds
- La puissance crête théorique de la machine est de 96 Gflops.

Un autre supercalculateur beaucoup plus puissant à architecture parallèle IBM RS6000/SP4 que nous utilisons est installé au Centre Informatique National de l'Enseignement Supérieur à Montpellier. Pour les chercheurs de toutes disciplines confrontés à des calculs intensifs et les demandeurs de puissance de calcul et de grandes capacités de mémoire, il constitue un super outil[20].

- 9 nœuds hauts performances, chacun comprenant :
- 1 nœud à 32 processeurs de connexion à base de Power4 à 1,3 GHz avec 64 Go de mémoire (2 Go par processeur)
- 1 autre nœud à 32 processeurs à base de Power4 à 1,3 GHz avec 64

Go de mémoire (2 Go par processeur)

- 2 nœuds à 32 processeurs à base de Power4+ à 1,7 GHz avec 64 Go de mémoire (2 Go par processeur)
- 5 nœuds à 32 processeurs à base de Power4+ à 1,7 GHz avec 32 Go de mémoire (1 Go par processeur)
- Les nœuds sont interconnectés par un réseau très haut débit HPS-Federation
- 4 To de disques (GPFS)
- La puissance crête théorique de la machine est de 1,85 T flops.

2.4 Description de GRID, P2P, XtremWeb,LAM-MPI

2.4.1 Description du GRID

Le GRID est une infrastructure du calcul et management des données qui fournit un fondement électronique à la société globale des affaires, du gouvernement, de la recherche, de la science et des divertissements [21]-[25]. Les GRIDs , illustrés dans FIG 2.3, intègrent le réseau, la communication, le calcul et les informations à fournir, c'est-à-dire un environnement virtuel pour le calcul et le management des données dans la même manière que l'Internet intègre les ressources à organiser une plate-forme virtuelle des informations. Le GRID est en train de transformer la science, les commerces, la santé et la société. L'infrastructure du GRID nous fournira toutes les ressources, avec les capacités de se relier dynamiquement comme un ensemble à supporter l'exécution à grande échelle, des ressources intensives et des applications distribuées [26].

Les GRIDs sont intrinsèquement distribués, hétérogènes et dynamiques. Ils permettent de profiter efficacement cycles et stockages infinis, également à accéder aux instruments, aux appareils de visualisation et les autres matériels sans tenir compte des localisations géographiques. Pour que cette réalisation soit réussie, il faut développer les systèmes complexes de logiciels et services qui permettent les accès conviviaux, utiliser efficacement ensemble les ressources, renforcer les protocoles qui coordonnent les ressources pour les utilisateurs d'une manière stable avec les meilleures performances.

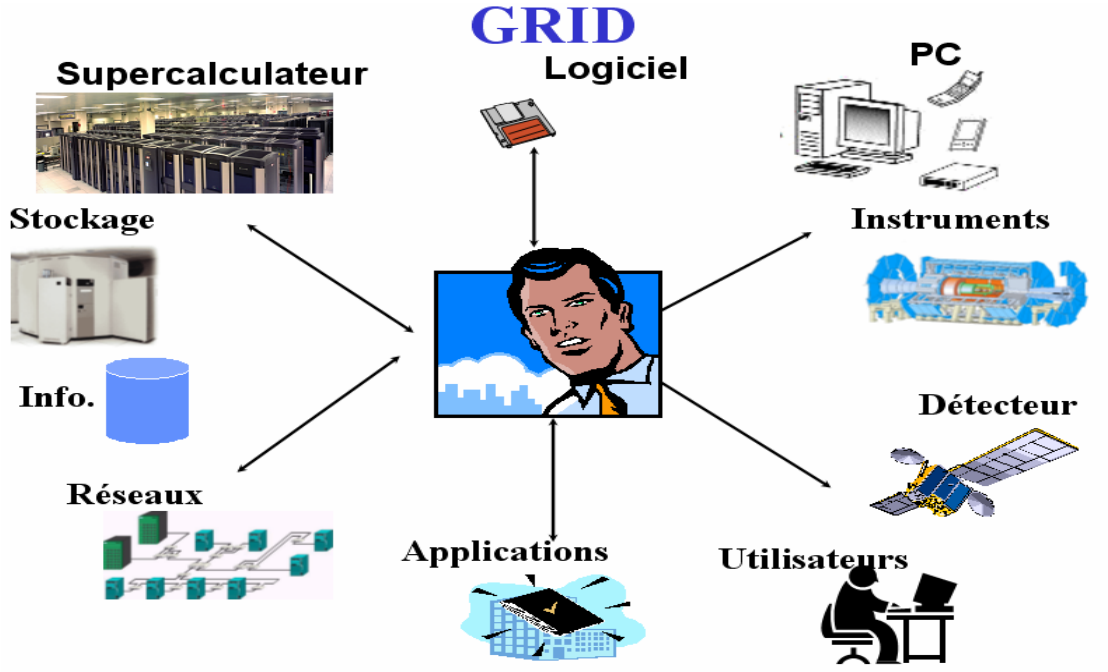


FIG 2.3 -L'architecture du GRID

2.4.2 Description du P2P

P2P est « peer to peer » en anglais. Communication de pair à pair, illustré dans FIG 2.4 : les deux machines qui communiquent sont sur un pied d'égalité, elles peuvent être toutes les deux client ou serveur, voire client et serveur à la fois,. C'est le mode de fonctionnement systématique sur l'Internet, mais la distinction client-serveur a provoqué un abus de langage : le sens commun a retenu que les postes de travail individuels sont uniquement des clients. Le P2P désigne alors le fait de transformer un tel poste en serveur de fichiers plus ou moins évolué. Exemples types de programmes implantant (ou ayant mis en œuvre) ce type de communication : Gnutella, Napster, eMule.

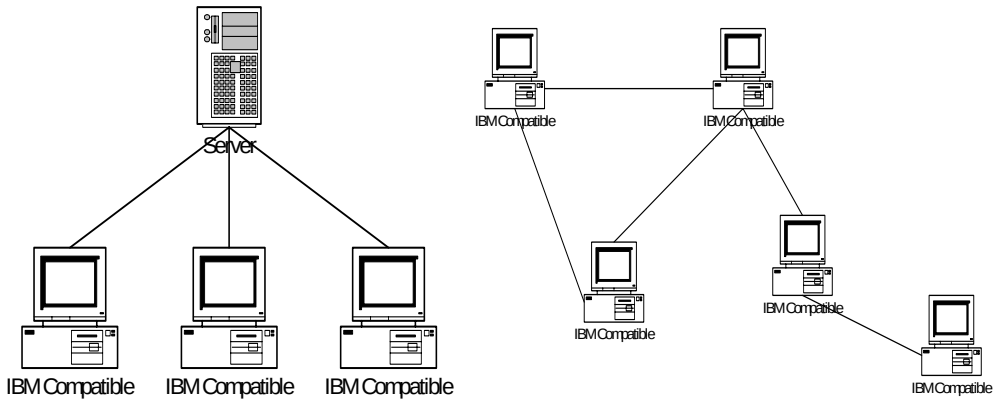


FIG 2.4-Comparaison entre la modèle centralisé et P2P

C'est une liaison poste à poste par opposition au modèle client-serveur. Dans ce type de réseau les ordinateurs sont connectés les uns aux autres sans passer par un serveur central. Ce type de réseau est très utilisé pour les échanges de fichiers pirates (MP3, Films, Images, Livres...), en effet il est impossible de mettre un terme à ces échanges car il n'y a pas de serveur central.

Dans tous les cas, un système Pair à Pair présume que toutes les machines coopèrent à une cible commune qui peut être un calcul ou le stockage/partage d'information.

2.4.3 Description du XtremWeb

Le but du projet XtremWeb est la conception et le développement d'un environnement de Calcul Global et de calcul Pair à Pair pluridisciplinaire et académique.

Le système XtremWeb est une plate-forme généralisée, sécurisée et visant de hautes performances de calcul. Ce système permet à des Universités, des Ecoles, des Institutions, et des Industriels d'installer et d'utiliser leur propre système de Calcul Global et calcul Pair à Pair pour réaliser leur travaux de recherche ou la production de calcul.

XtremWeb a réussi à atteindre l'un des objectifs de la plate-forme : Il impose le moins de contraintes possibles à l'environnement à mettre en œuvre pour les travaux de calcul. Il est une plate-forme généraliste, parce qu'il se distingue des systèmes comme SETI@home qui sont dédiés à une application spéciale. Il est aussi utilisable pour une classe d'applications.

Ce système n'est fait qu'à partir de standards éprouvés et de logiciels libres de type open source comme les langages C++, Java, le langage de scripts PERL, le protocole XML RPC, le serveur de Web Apache et la base de données MySQL. Les utilisateurs d'XtremWeb peuvent télécharger les codes de sources, accéder facilement à tous les logiciels.

L'objectif de hautes performances d'XtremWeb signifie que nous visons les applications du calcul intensif. Mais la plate-forme XtremWeb n'est comparable directement avec les supercalculateurs listés dans le top 500. En effet cette comparaison est impossible, car d'abord le nombre de ressources utilisables est beaucoup plus grand pour un systèmes de Calcul Global, les PCs avec les systèmes d'exploitation différents, les clusters, voire les appareils mobiles, PDAs, les téléphones mobiles, etc. puis les ressources sont essentiellement volatiles (elles peuvent se déconnecter à tout moment) et les performances de communications sont très faibles [26].

2.4.4 LAM-MPI environnement de calcul

LAM-MPI (Local Area Multi-computer MPI) est un environnement de calcul MPI et un système de développement pour les ordinateurs hétérogènes dans un réseau [28]. MPI communication standard est, en fait, le plus haut niveau du logiciel Trollius. La librairie de communication de LAM-MPI est divisé en trois parties : la couche MPI, le Request Progression Interface (RPI) [29], et le noyau du Trollius.

La fonction principale de la couche MPI est de vérifier des paramètres, d'exécuter les tâches « bookkeeping » et appeler le RPI ou les fonctions du Trollius pour les communications des données. Toutes les communications des envois et réceptions à l'intérieur de LAM - MPI sont organisées comme des

requêtes, de façon à ce que leur gestion uniforme soit possible. Quand la couche MPI génère une requête, et si la file d'attente est vide, cette requête est passée directement au RPI et exécutée immédiatement. Si la file n'est pas vide, la nouvelle requête est marquée comme bloquante et mise à la file. Puis le RPI traite la file, et ne retournera que l'indication que le message est entièrement envoyé. Une fois le RPI a fini une requête, il marque la requête comme complète et la couche MPI l'a mise hors de la file. A présent, LAM-MPI inclut l'implantation TCP (utiliser IDS-internet domain socket entre les processeurs), l'implantation USYSV (la même que le TCP sauf que la mémoire partagée est utilisée pour les communications entre les processeurs du même nœud), et l'implantation SYSV (identique à USYSV sauf que le SYSV utilise des sémaphores pour verrouiller la mémoire partagée entre les processeurs)

2.5 Conclusion

Ce chapitre a présenté plusieurs systèmes parallèles et distribués visant à résoudre les systèmes linéaires à grande échelle. Nous présentons les trois types de machines parallèles SIMD, SPMD et MIMD. En outre, deux systèmes de supercalculateur IBM SP3 et IBM SP4 que l'on utilise pour nos tests numériques sont également présentés.

Les brèves introductions des conceptions du GRID et de P2P sont aussi présentées pour donner une idée générale du calcul de grille et de supercalculateur. Deux systèmes de grille de calcul XtremWeb et LAM-MPI sont également décrits dans ce chapitre.

Dans les chapitres suivants, les tests, leurs résultats et analyses avancées de l'implantation de la méthode hybride GMRES(***m***)/LS-Arnoldi sur la plate-forme de supercalculateurs sont détaillés (Chapitre 4 et Chapitre 5). Nous réalisons aussi l'algorithme GMRES(***m***) sur les plates-formes de grille XtremWeb et LAM-MPI , puis nous montrons les analyses et nous comparons les performances obtenues (Chapitre 6).

Chapitre 3 Résolution de systèmes linéaires non symétriques de très grande taille

De nombreux applications scientifiques ou industrielles nécessitent la résolution de systèmes linéaires non symétriques, de très grande taille. Cette résolution demande souvent d'importants temps de calculs, comparés au traitement du reste du problème. L'optimisation de ce type de calcul est donc capitale et la grande taille de la structure de données implique le recours au parallélisme et à des méthodes appropriées.

Les systèmes linéaires concernés sont généralement exprimés sous la forme de matrices, la plupart du temps très creuses, de la spécificité desquelles il convient de tenir compte aussi bien pour la représentation en mémoire que pour le choix des méthodes de calcul.

3.1 Matrices creuses

3.1.1 Intérêt

Beaucoup de matrices de grande taille de problèmes industriels comprennent, un très grand nombre d'éléments nuls, et ils sont alors des matrices « creuses ». La prise en compte du creux dans l'algorithme est fondamentale en terme d'efficacité. En fait, il est vain d'allouer une place en mémoire aux éléments nuls, ainsi qu'il est inutile d'exécuter les calculs les concernant. En utilisant les formats corrects, on peut donc espérer gagner de la place en mémoire et du temps de calcul.

Il est donc nécessaire de compresser les matrices creuses, c'est-à-dire trouver un format qui représente ces matrices[8],[9],[20] qui permet :

- ✓ de ne pas enregistrer les éléments nuls,
- ✓ de retrouver aisément un élément connaissant ses coordonnées,

- ✓ d'exécuter facilement les opérations courantes sur les matrices (notamment le calcul de la transposée).

3.1.2 Formats

Format CSR

Le format le plus économe en mémoire est le format **CSR** (Compress Sparse Row) ou son équivalence par colonnes **CSC** (Compress Sparse Column). Le format **CSR** décrit la matrice creuse par trois vecteurs (voir l'exemple FIG 3.1)

$$\begin{pmatrix} 1 & 0 & 8 & 0 & 3 & 0 & -2 & 0 \\ 0 & 2 & 0 & 0 & 8 & 0 & 0 & 6 \\ 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 \\ 0 & -8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 & 8 \end{pmatrix} \xrightarrow{compression} \begin{Bmatrix} 1 & 8 & 3 & -2 \\ 2 & 8 & 6 & \\ 5 & & & \\ -8 & & & \\ 1 & & & \\ 1 & & & \\ 5 & & & \\ 4 & 8 & & \end{Bmatrix}$$

Les Vecteurs compresses au format CSR :

$$A = \{1 \quad 8 \quad 3 \quad -2 \quad 2 \quad 8 \quad 6 \quad 5 \quad -8 \quad 1 \quad 1 \quad 5 \quad 4 \quad 8\}$$

$$JA = \{1 \quad 3 \quad 5 \quad 7 \quad 2 \quad 5 \quad 8 \quad 4 \quad 2 \quad 4 \quad 6 \quad 4 \quad 1 \quad 8\}$$

$$IA = \{1 \quad 5 \quad 8 \quad 9 \quad 10 \quad 11 \quad 12 \quad 13 \quad 15\}$$

FIG 3.1-Exemple de compression d'une matrice creuse au format CSR
(compression et concaténation des lignes)

A est un vecteur contenant tous les éléments non nuls de la matrice, rangés ligne par ligne.

JA est un vecteur de même taille que **A** dont chaque élément contient le numéro de colonne de l'élément correspondant de **A**.

IA est un vecteur de taille $n+1$, où n est le nombre de lignes de la matrice. Chaque élément IA(i) de ce vecteur ($i \in [1, n]$) contient le rang, dans le vecteur A, du 1^{er} élément de la ligne n°i de la matrice.

IA($n+1$) contient T+1, où T est la taille de A (et de JA).

L'accès à l'élément M (l, c) de la matrice M se fera donc de la manière suivante :

- Recherche des indices de début et de fin de la ligne n°l :

$$\begin{cases} \text{début=IA(l)} \\ \text{fin= IA(l+1)-1} \end{cases}$$

- Recherche de la valeur c dans **JA** entre **JA** (*début*) et **JA** (*fin*) (Cette recherche peut être obtenue en complexité logarithmique).

- Si c est trouvé à la position **JA**(i), le partie réelle d'élément cherché est **A**(i); si c n'est pas trouvé, l'élément cherché est 0.

Format Ellpack-Itpack

Le format Ellpack-Itpack procède aussi à une compression des lignes. Il utilise deux tableaux de dimensions (1 :n, 1 :LMAX), où n est le nombre de lignes de la matrice d'origine, et LMAX le nombre maximum d'éléments non nuls dans une ligne de cette matrice (voir l'exemple de FIG 3.2*)*

$$\begin{pmatrix} 1 & 0 & 8 & 0 & 3 & 0 & -2 & 0 \\ 0 & 2 & 0 & 0 & 8 & 0 & 0 & 6 \\ 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 \\ 0 & -8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 & 8 \end{pmatrix} \xrightarrow{compression} \left\{ \begin{array}{cccc} 1 & 8 & 3 & -2 \\ 2 & 8 & 6 & \\ 5 & & & \\ -8 & & & \\ 1 & & & \\ 1 & & & \\ 5 & & & \\ 4 & 8 & & \end{array} \right\}$$

Les matrices au format Ellpack – Itapack :

$$A = \left\{ \begin{array}{cccc} 1 & 8 & 3 & -2 \\ 2 & 8 & 6 & \\ 5 & & & \\ -8 & & & \\ 1 & & & \\ 1 & & & \\ 5 & & & \\ 4 & 8 & & \end{array} \right\} \qquad JA = \left\{ \begin{array}{cccc} 1 & 3 & 5 & 7 \\ 2 & 5 & 8 & \\ 4 & & & \\ 2 & & & \\ 4 & & & \\ 6 & & & \\ 4 & & & \\ 1 & 8 & & \end{array} \right\}$$

FIG 3.2-Exemple de compression d'une matrice creuse au format Ellpack-Itpack (compression des lignes)

A est un tableau dont chaque ligne contient les éléments non nuls d'une ligne de la matrice.

JA est un tableau dont chaque élément contient le numéro de colonne de l'élément correspondant de **A**.

L'accès à l'élément M(*l*,*c*) de la matrice M, légèrement plus rapide que dans le format précédent, se fera de la façon suivante :
Recherche de la valeur *c* dans la ligne JA(*l*) (Cette recherche peut être obtenue en complexité logarithmique).

Si *c* est trouvé à la position JA(*i*), l'élément cherché est A(i) ; si c n'est pas trouvé, l'élément cherché est 0.

Ce format, plus gourmand en mémoire que le précédent (car certaines lignes contiennent moins de LMAX éléments non nuls), est intéressant car il permet un découpage aisé de la matrice entre plusieurs processus parallèles. Il est notamment bien adapté à une parallélisation SPMD à gros grain.

3.1.3 Parallélisation

Format Ellpack-Itpack

La distribution de la structure de données sur les différents processeurs est particulièrement simple pour une matrice creuse compressée au format Ellpack-Itpack. Il suffit, en fait, de découper les tableaux **A** et **JA** en bandes horizontales de hauteur identique, chaque bande étant alloué à un processeur.

$$
\begin{array}{llll}
A_1=\{\ldots\ \ldots\ \ldots\} & JA_1=\{\ldots\ \ldots\ \ldots\} & IA_1=\{\ldots\ \ldots\ \ldots\} & \rightarrow CPU\,n°1 \\
A_2=\{\ldots\ \ldots\ \ldots\} & JA_2=\{\ldots\ \ldots\ \ldots\} & IA_2=\{\ldots\ \ldots\ \ldots\} & \rightarrow CPU\,n°2 \\
\vdots & \vdots & \vdots & \vdots \\
A_i=\{\ldots\ \ldots\ \ldots\} & JA_i=\{\ldots\ \ldots\ \ldots\} & IA_i=\{\ldots\ \ldots\ \ldots\} & \rightarrow CPU\,n°i \\
\vdots & \vdots & \vdots & \vdots \\
A_p=\{\ldots\ \ldots\ \ldots\} & JA_p=\{\ldots\ \ldots\ \ldots\} & IA_p=\{\ldots\ \ldots\ \ldots\} & \rightarrow CPU\,n°p
\end{array}
$$

FIG 3.3-Distribution des données d'une matrice creuse au format CSR sur une machine SPMD

Nous pouvons remarquer que dans cette distribution, chaque ligne est totalement allouée à un seul processeur, alors que chaque colonne est distribuée sur l'ensemble des ***p*** processeurs, Il s'ensuit que chaque fois qu'une colonne sera indispensable au calcul achevé par l'un des processeurs(par exemple dans un

produit), les morceaux absents de la colonne devront être envoyés par les ***p-1*** autres processeurs, ce qui provoque un nombre de communications important : ***p(p-1)*** communications, contenant chacune un morceau de vecteur de longueur $\frac{n}{p}$, sont obligatoires ici pour un produit matrice-vecteur. Or le produit matrice-vecteur apparaît fréquemment [10] dans la majorité des méthodes itératives pour les systèmes linéaires, et notamment dans l'algorithme GMRES.

Formats Ellpack-Itpack et CSR

Nous pouvons combiner la facilité de découpage de la matrice au format Ellpack-Itpack avec l'économie de place en mémoire fournie par le format CSR en hybridant ces deux formats de la manière suivante :

La matrice est générée au format Ellpack-Itpack, et remplie dans ce format par un processus d'initialisation. Ce processus découpe la matrice en bandes destinées aux différents processus de calcul.

Chaque bande est convertie au format CSR à l'instant de l'envoi des données aux processus de calcul (voir FIG 3.3). En plus, cette manière offre l'atout de réduire la durée de l'initialisation, le volume des données à transmettre étant limité (le coût de la conversion est insignifiant).

3.2 Méthodes de résolution

Pour résoudre un système linéaire, il existe nombreuses méthodes.

Pour les systèmes présentés par des matrices creuses de grandes tailles, les méthodes directes (Gauss, Jordan, LU, …) ne sont pas idéales. En fait, elles changent, pendant le calcul, la structure creuse de la matrice. Par exemple, la méthode du pivot de Gauss, commence par mettre la matrice sous forme triangulaire supérieure. A chaque étape de ce calcul, on fait apparaître des zéros dans le bas d'une colonne, par un produit matriciel. Celui-ci peut faire apparaître, à la place d'un élément nul de la matrice initiale, un élément calculé non nul (voir l'exemple de *FIG 3.4*). Il existe des méthodes limitant le remplissage mais elles sont

relativement complexes à réaliser, particulièrement dans un contexte donnant plusieurs prototypes d'exécutions parallèles.

En outre, à cause à la limite de précision liée à la machine elles supportent assez mal les arrondis. Ces imprécisions peuvent s'accumuler au cours des calculs, causant des erreurs importantes à la fin de la résolution du système. Evidemment, quand on augmente la taille de la matrice ces problèmes sont pires. Il faut modifier la structure compressée, pour que ce nouvel élément soit inséré, ce qui peut être onéreux.

Les méthodes itératives permettent fréquemment de conserver la structure creuse de la matrice, mais il faut souvent choisir des nombreux paramètres délicats. Le principe de ces méthodes est, par un calcul itératif, à partir d'un vecteur initial x_0 plus ou moins aléatoire, de développer une série de vecteurs $\{x_0, x_1, ..., x_k\}$ convergeant vers le vecteur résultat $\bar{x}$, en calculant à chaque itération i le résidu $b - Ax$. Ces méthodes sont beaucoup moins sensibles aux problèmes d'arrondis, et ne modifient pas la matrice initiale A. Elles conservent ainsi la structure creuse et profitent de ses avantages. Il convient de veiller à ce que ces méthodes convergent le plus rapidement possible. Pour ces méthodes certains problèmes de stabilité numérique se posent également, même si leurs traitements peuvent parfois être plus simples que dans le cas des méthodes directes.

Matrice creuse:(le vecteur second membre est omis)

$$A=\begin{pmatrix}-1&1&0&4&0&0\\0&7&0&0&0&0\\6&0&-2&0&1&4\\0&0&0&2&0&0\\0&5&0&0&1&3\\1&1&0&0&0&5\end{pmatrix}\quad\rightarrow\quad A'=\begin{pmatrix}-1&1&0&4&0&0\\0&7&0&0&0&0\\0&\mathbf{6}&-2&\mathbf{24}&1&4\\0&0&0&2&0&0\\0&5&0&0&1&3\\0&2&0&\mathbf{4}&0&5\end{pmatrix}$$

Format CSR :

$A=\{-1\ \ 1\ \ 4\ \ 7\ \ 6\ \ -2\ \ 1\ \ 4\ \ 2\ \ 5\ \ 1\ \ 3\ \ 1\ \ 1\ \ 5\}$

$A'=\{-1\ \ 1\ \ 4\ \ 7\ \ \mathbf{6}\ \ -2\ \ \mathbf{24}\ \ 1\ \ 4\ \ 2\ \ 5\ \ 1\ \ 3\ \ 2\ \ \mathbf{4}\ \ 5\}$

$JA=\{1\ \ 2\ \ 4\ \ 2\ \ 1\ \ 3\ \ 5\ \ 6\ \ 4\ \ 2\ \ 5\ \ 6\ \ 1\ \ 2\ \ 6\}$

$JA'=\{1\ \ 2\ \ 4\ \ 2\ \ \mathbf{2}\ \ 3\ \ \mathbf{4}\ \ 5\ \ 6\ \ 4\ \ 2\ \ 5\ \ 6\ \ 2\ \ \mathbf{4}\ \ 6\}$

$IA=\{1\ \ 4\ \ 5\ \ 9\ \ 10\ \ 13\ \ 15\}$

$IA'=\{1\ \ 4\ \ 5\ \ 10\ \ 11\ \ 14\ \ 17\}$

FIG 3.4- La modification de la structure creuse e par la méthode de Gauss après le traitement de la première colonne

La méthode GMRES (m), depuis son introduction, est la méthode itérative très souvent utilisée pour résoudre des systèmes linéaires présentés par des matrices creuses non symétriques et de grande taille.

ALGORITHME 1 : GMRES

1. Choisir la solution initiale x_0 ,

   Calculer $r_0=b-Ax_0$, et $v_1=\dfrac{r_0}{\|r_0\|_2}$

2. Répéter pour $j=1,...,k,...,$

$$h_{i,j}=(Av_j,v_i),i=1\cdots j$$

$$\hat{v}_{j+1}=Av_j-\sum_{i=1}^{j}h_{i,j}v_i$$

$$h_{j+1,j}=\|\hat{v}_{j+1}\|_2$$

$$v_{j+1}=\frac{\hat{v}_{j+1}}{h_{j+1,j}}$$

3. Etablir la solution approchée :

   $x_k=x_0+V_k y_k$, où y_k est solution de l'expression :

$$\min_{y\in\Re^k}\|\beta e_1-\overline{H}_k y\|_2$$

FIG 3.5 - Algorithme n°1 : La méthode GMRES

3.2.1 La méthode GMRES(m)

Principe

Y. Saad et M. H. Schultz ont présenté la méthode GMRES en 1986. GMRES est l'abréviation de « **G**eneralized **M**inimum **RES**idual ». Cette méthode itérative permet de résoudre efficacement les systèmes linéaires non symétriques de grande taille de type :

$$Ax = b \tag{3.1}$$

La minimisation des résidus successivement acquis pour chaque itéré est la base de cette méthode:

$$r_i = Ax_i - b \tag{3.1*}$$

où ***A*** représente une matrice réelle, ***b*** le vecteur second membre, et ***x*** le vecteur solution. r_i le vecteur résiduel. Sans modifier la structure de la matrice, cette méthode est notamment bien adaptée à la résolution des systèmes décrits par des matrices creuses.

Il s'agit d'une méthode de projection, basée sur l'algorithme d'Arnoldi. Celui-ci utilise la méthode de Gram-Schmidt pour calculer une base orthonormée $\{v_1, v_2, \cdots, v_k\}$ du sous-espace de Krylov $K_k = span\{v_1, Av_1, \cdots, A^{k-1}v_1\}$. Nous obtenons ainsi la matrice de Hessenberg $H_k = V_k^T AV_k$ de taille *k,* et la matrice $\overline{H}_k$, construite à partir de ***H*** en lui ajoutant une ligne supplémentaire dont le seul élément non nul est $h_{k+1,k}$, à la position **(***k+1, k***)**. Les vecteurs v_i et la matrice $\overline{H}_k$ satisfont la relation suivante :

$$AV_k = V_{k+1}\overline{H}_k \tag{3.2}$$

ALGORITHME 2 : GMRES(***m***)

1. Choisir la solution initiale x_0 ,

   Calculer $r_0 = b - Ax_0$, et $v_1 = \frac{r_0}{\|r_0\|_2}$

2. Répéter pour $j = 1,...,m$

   $h_{i,j} = (Av_j, v_i), i = 1 \cdots j$

   $\hat{v}_{j+1} = Av_j - \sum_{i=1}^{j} h_{i,j} v_i$

   $h_{j+1,j} = \left\| \hat{v}_{j+1} \right\|_2$

   $v_{j+1} = \frac{\hat{v}_{j+1}}{h_{j+1,j}}$

3. Etablir la solution approchée :

   $x_m = x_0 + V_m y_m$, où y_m minimise $\left\| \beta e_1 - \overline{H}_m y \right\|_2$, $y \in \Re^m$,

4. Redémarrage :

   Calculer $r_m = b - Ax_m$

   Si r_m est satisfaisant alors arrêter,

   Sinon réinitialiser $x_0 = x_m$, $r_0 = r_m$, $\beta = \left\| r_0 \right\|_2$ et $v_1 = \frac{r_0}{\beta}$ retourner à l'étape n°2

FIG 3.6-Algorithme n°2 : La méthode GMRES(m)

Y. Saad et M.H.Schultz ont présenté dans [1] que si x_0 est la solution initiale du système, la solution approchée x_k à l'étape

$$x_k = x_0 + V_k y_k \tag{3.3}$$

peut être trouvée en calculant y_k qui est la solution du problème des moindres carrés

$$\min_{y \in \Re^k} \left\| \beta e_1 - \overline{H}_k y \right\|_2 \tag{3.4}$$

où β est la norme résiduelle, c'est-à-dire $\beta = b - A x_0$,et e_1 est le premier vecteur canonique de $\Re^{k+1}$. L'algorithme GMRES se présente alors conformément à la FIG 3.5*(algorithme n°1).*

La valeur de *k* permet d'obtenir la solution avec une précision suffisante qui peut être très grande, il signifie qu'il faut conserver en mémoire un nombre de

vecteurs proportionnels à *k* et d'exécuter un nombre important de multiplications (**A***v*) en $O(k^2)$. Pour remédier à ces désavantages, Y.Saad et M.H.Schultz ont formulé l'algorithme GMRES (*m*), Selon cet algorithme, on procède un redémarrage toutes les *m* étapes, *m* est un paramètre constant défini par l'utilisateur lui-même (voir FIG 3.6, algorithme n°2).

Pour résoudre le problème des moindres carrés l'application des transformations de Householder permet d'obtenir une stabilité satisfaisante [11].

3.2.2 Degré de parallélisme limité de la méthode GMRES(m)

Les processus de GMRES montrent un accouplement fort, ainsi les communications sont intenses. Quand nous augmentons la granularité de cette méthode avec davantage de processeurs participants, les performances deviennent mauvaises après avoir dépassé un certain seuil. Il existe un nombre optimal de processeurs. Au delà de ce nombre, le temps permettant d'atteindre la convergence augmente.

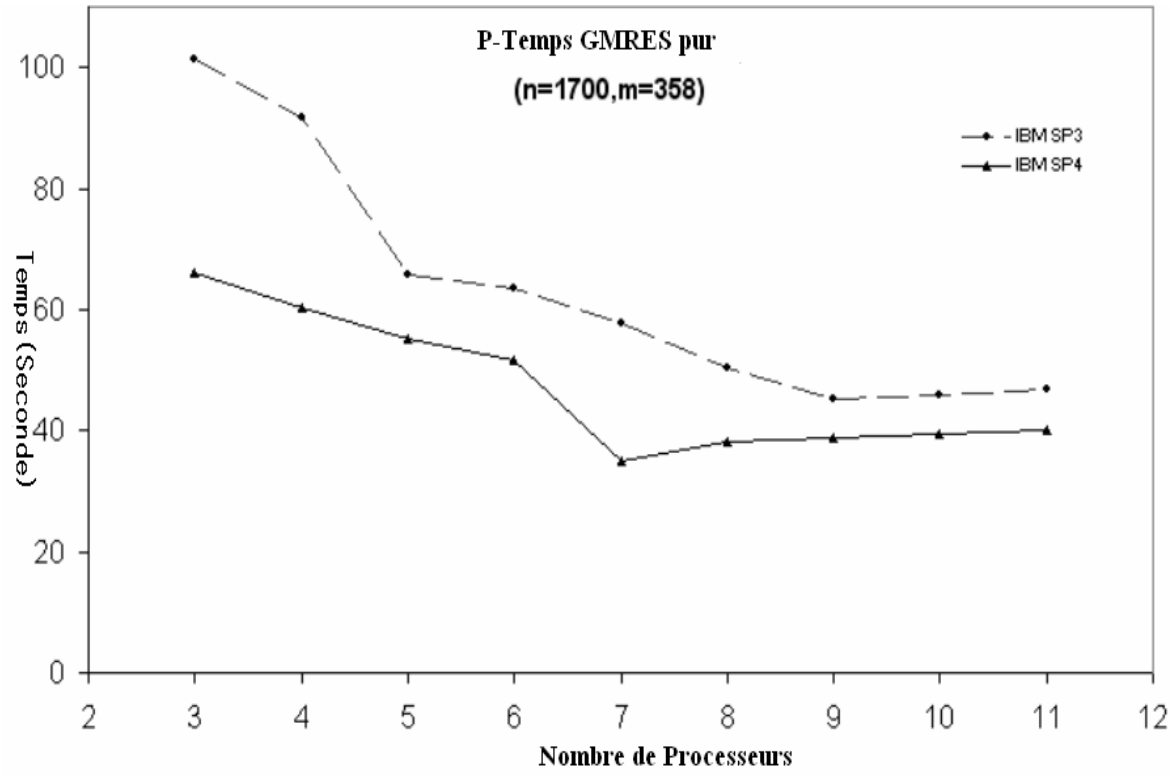


FIG 3.7 P-Temps de GMRES pur de la matrice « utm1700a »

Dans la *FIG 3.7*, nous remarquons que 9 est le nombre optimal pour cette matrice « utm1700a » et sur la machine IBM SP3. Toutefois pour IBM SP4 le nombre

optimal de processeurs pour la même matrice est 7. Pour cet algorithme, IBM SP4 obtient une meilleure performance qu'IBM SP3 grâce à une meilleure configuration de matériel. Quand nous augmentons le nombre de processeurs à 10 ou 11 processeurs, le temps pour atteindre la convergence n'augmente pas beaucoup grâce au mécanisme de la mémoire partagée dans juste un noeud d'IBM SP3 ou de SP4 (le coût de communication est beaucoup plus lourd par le message passant que le mode de mémoire partagée).

3.2.3 La méthode hybride GMRES/LS-Arnoldi

Principe

Certaines matrices ont besoins d'un grand nombre d'itérations, par la méthode GMRES, avant d'atteindre la convergence. C'est sur ce nombre que la méthode hybride va jouer, en cherchant à améliorer le vecteur initial x_0 du redémarrage à l'aide de la méthode « Least Square » utilisant la connaissance des valeurs propres de la matrice ; du moins de certaines d'entre elles car leur calcul exhaustif, si on savait le faire efficacement, durerait souvent plus longtemps que la résolution du système linéaire, objet de notre problème.

Nous allons donc employer les nœuds de notre machine parallèle inutilisés par GMRES, ou bien une deuxième machine parallèle reliée à notre réseau comme nous le faisions auparavant, pour calculer un certain nombre de valeurs propres de la matrice par la méthode d'Arnoldi. Celles-ci seront ensuite utilisées par la méthode « Least Square » pour construire une suite d'itérés qui permettront de proposer un meilleur vecteur initial pour redémarrer la méthode GMRES(***m***) [19].

Méthode d'Arnoldi

La méthode d'Arnoldi [12][13][14] est une méthode de projection sur un sous-espace de Krylov. Elle permet de calculer de manière itérative une bonne approximation d'un certain nombre de valeurs propres.

L'algorithme de base, nommé « projection d'Arnoldi », employé dans cette méthode, a d'ailleurs été repris comme un composant de la méthode GMRES (voir les algorithmes 1 et 2,FIG 3.5 et FIG 3.6). Cette projection de la matrice ***A*** de taille *n* sur l'espace de Krylov $K_m(A,v)$ permet d'obtenir une matrice de Hessenberg supérieure H_m de taille *m*. Les valeurs propres de H_m sont baptisées les valeurs de Ritz de ***A,*** elle sont les approximations des valeurs propres correspondantes de ***A.***

La qualité de cette approximation augmente avec *m* [14], mais l'augmentation de la taille de H_m amplifie en même temps le coût des calculs (en $O(m^2)$) et la taille mémoire nécessaire (*m* vecteurs de taille *n,* en plus de la matrice H_m). Pour remédier à ces inconvénients, nous allons également mettre en place une stratégie de redémarrage, conservant une valeur de *m* suffisamment faible, combinaison linéaire de la partie réelle des vecteurs propres associés à un certain nombre de valeurs propres de plus grands modules, le vecteur *v* sera utilisé comme vecteur initial pour ce redémarrage. L'algorithme avec redémarrage de la méthode d'Arnoldi est décrit sur la FIG 3.8*(algorithme n°3).*

ALGORITHME 3 : METHODE D'ARNOLDI

1. Choisir un vecteur initial v de norme 1, la dimension m du sous-espace de Krylov, le nombre désiré d de valeurs propres de plus grand module, avec la précision ε_α .
2. Effectuer la projection d'Arnoldi :
   Répéter pour $j = 1, \cdots, m$
   $h_{i,j} = (Av_j, v_i), i = 1 \cdots j$
   $w_j = Av_j - \sum_{i=1}^{j} h_{i,j} v_i$
   $h_{j+1,j} = \|w_j\|_2$ Si $h_{j+1,j} = 0$ alors arrêter
   $v_{j+1} = \frac{w_j}{h_{j+1,j}}$
3. Calculer les valeurs propres ($\lambda_i, 1 \le i \le d$) et les vecteurs propres associés ($y_i, 1 \le i \le d$) de H_m
4. Calculer les vecteurs de Ritz $u_i = V_m y_i$ pour $i = 1, \cdots, d$
5. Calculer les normes résiduelles $\rho_i = \|\lambda_i u_i - A u_i\|_2 (1 \le i \le d)$
6. Redémarrage :
   Si $\max_{i=1}^{d} |\rho_i| < \varepsilon_\alpha$ alors arrêter
   Sinon établir $v = \sum_{i=1}^{d} \mathrm{Re}(u_i)$ et aller à l'étape *n°2*

FIG 3.8–Algorithme n°3 : La méthode d'Arnoldi

Méthode des moindres carrées (Least Squares)

La méthode « Least Squares » [15] est une méthode itérative dans laquelle l'itéré $\tilde{x}$ est défini par

$$\tilde{x} = x_0 + P_k(A) r_0 \tag{3.5}$$

où x_0 est le vecteur initial, r_0 son résidu ($r_0 = b - Ax_0$), et P_k un polynôme de degré $k-1$.

A partir des coordonnées des sommets du polygone convexe qui englobent les valeurs propres de ***A*** représentées dans le plan complexe, ainsi qu'à partir des paramètres de l'ellipse circonscrite à ce convexe nous calculons ce polynôme (voir FIG 3.9). Cette ellipse présente une symétrie par rapport à l'axe des réels, à condition que la matrice ***A*** elle-même soit réelle. Quand ***A*** est complexe, nous

nous heurtons une difficulté bien plus importante de la méthode « Least Squares », qui sera étudiée dans le Chapitre 5.

Il serait possible de trouver la solution du système linéaire par l'emploi de la seule méthode « Least Square », mais la convergence ne sera arrivée qu'avec l'obtention d'un nombre important de valeurs propres, nécessitant un calcul lourd. Donc ici nous tâchons d'utiliser les propriétés de cette méthode afin d'accélérer la méthode GMRES(m). Le but est de prendre ce qu'il y a de mieux dans chacune des méthodes pour calculer au plus vite la solution.

Dans la méthode hybride proposée, la méthode « Least Squares » ne sera appliquée que ponctuellement, un nombre déterminé l d'itérations de ce type n'ayant lieu que quand un nombre suffisant des valeurs propres est disponible, celles-ci étant calculées avec l'aide de la méthode d'Arnoldi, cf. l'algorithme 3 (FIG 3.9). L'algorithme hybride sera donc celui de FIG 3.10 (algorithme n°4 dans lequel P_k est calculé séparément dès que nous avons obtenu les valeurs propres suffisantes).

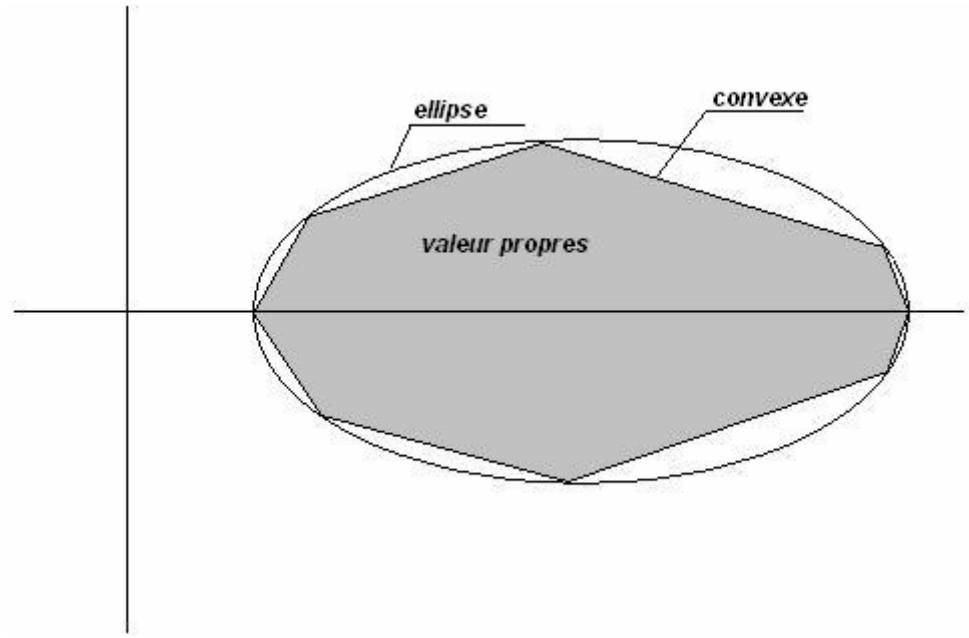


FIG 3.9– Représentation dans le plan complexe des valeurs propres d'une matrice réelle, ainsi que du convexe et de l'ellipse les englobant

Si le convexe calculé contient uniquement les valeurs propres $\lambda_1, \cdots, \lambda_s$, alors le dernier résidu respectera la relation [19] :

$$\tilde{r} = (R_k(A))^l r_0 \tag{3.6}$$

$$= \sum_{i=1}^{s} \rho_i (R_k(\lambda_i))^l u_i + \sum_{i=s+1}^{n} \rho_i (R_k(\lambda_i))^l u_i \tag{3.7}$$

Puisque la méthode « Least Squares » trouve R_k qui minimise $|R_k(\lambda)|$ (λ faisant partie du convexe obtenu), la première partie du résidu est très réduite. Mais pour la seconde partie, le résidu dépendra beaucoup des vecteurs propres associés aux valeurs propres qui se situent en dehors du convexe. Ainsi, quand *l* croît, la première partie s'approche de zéro alors que la seconde partie devient très grande, ce qui explique le fait que le résidu puisse subir une augmentation très importante.

En redémarrant GMRES(*m*) avec un itéré dont le résidu est le résultat de plusieurs itérations de la méthode « Least Square », les composantes de ce nouveau vecteur sur le sous-espace propre lié aux valeurs propres prises en compte sont donc très petites. Or, c'est justement sur ce sous-espace, ou un sous-espace proche incluant en général celui-là, que la méthode GMRES(*m*) va minimiser le résidu adéquat. Néanmoins, les autres composantes du nouveau vecteur initial peuvent être très grandes et le résidu lié à GMRES devient très grand lors de première itération. Mais, dès que la méthode GMRES minimise le résidu sur le sous-espace de Krylov adéquat, les propriétés du nouveau vecteur initial vont permettre une réduction rapide du résidu.

De même, il peut être meilleur d'utiliser le résidu calculé après l'application de la méthode « Least Square » comme vecteur initial pour les redémarrages de la méthode d'Arnoldi afin de découvrir de nouvelles valeurs propres en dehors du convexe calculé antérieurement [16]. Cette technique est exploitée dans la méthode « Least Squares-Arnoldi » [14].

Principaux paramètres de la méthode hybride GMRES/LS-Arnoldi

Les paramètres suivants (liste non exhaustive) ont une grande influence sur les performances acquises par ces méthodes numériques, spécialement de la méthode hybride :

n est la taille de la matrice ***A.***

m est la taille du sous-espace de Krylov. On l'utilise non seulement dans la méthode GMRES qui solutionne le système linéaire mais aussi dans la méthode d'Arnoldi qui calcule les valeurs propres. Quand on diminue la valeur de *m*, la durée de chaque itération diminue, elle aussi. Néanmoins le nombre d'itérations nécessaires à la convergence augmente. On pourra mettre en évidence une valeur optimale de ce paramètre. Pour ces deux algorithmes, elle peut être différente. En outre, on peut profiter de ces deux valeurs de *m* pour régler la vitesse relative des deux calculs, afin d'ajuster le nombre d'envois de valeurs propres pour une efficacité optimale. Il faudra donc distinguer *m***(GMRES)** de *m***(Arnoldi)**.

k est le degré du polynôme « Least Squares »

l est la puissance du calcul « Least Squares ». Ce paramètre est le nombre d'itérations « Least Squares » exécutées à chaque prise en compte d'une nouvelle série de valeur propres supplémentaires.

ALGORITHME 4 : GMRES(*m*)/LS(*k*,*l*)-Arnoldi(m2) [Partie GMRES/LS]

1. Choisir le vecteur initial x_0 , la dimension *m* du sous-espace de Krylov pour GMRES, *k* le degré du polynôme Least Squares, *l* le nombre d'itérations Least Squares successives.
2. Calculer x_m , le $m^{ième}$ itéré de GMRES démarré avec x_0 .

   Faire $x_0 = x_m$ et $r_0 = b - Ax_0$

   Si $\|r_0\|_2 < \varepsilon_g$ alors arrêter
3. Si un nouveau polynôme P_k est disponible,

   alors faire l'étape n°4 (Least Squares)

   Sinon

   aller à l'étape n°2(GMRES)

   fin si
4. Pour $i = 1, \cdots, l$

   calculer $\tilde{x} = x_0 + P_k(A)r_0$

   faire $x_0 = \tilde{x}$ et $r_0 = b - Ax_0$

   fin pour

   Si $\|r_0\|_2 < \varepsilon_g$ alors

   arrêter

   sinon

   aller à l'étape n°2 (GMRES)

   fin si

FIG 3.10–Algorithme n°4 : La méthode hybride GMRES (m)/LS (k, l) -*Arnoldi(m2)*

3.3 Conclusion

Dans ce chapitre, nous présentons les formats utilisés pour mettre en œuvre notre algorithme parallèle avec des matrices de très grande taille. En effet, dans le contexte des grilles de calcul et de parallélisation, il faut employer des formats efficaces pour économiser les mémoires partagées ou distribuées et diminuer les communications au sein du réseau local ou à distance.

L'évolution des algorithmes de la méthode GMRES à la méthode hybride GMRES(***m***)/LS-Arnoldi est également expliquée dans ce chapitre. La méthode hybride comprend 3 parties principales : l'algorithme « GMRES(***m***) », la méthode « Least Squares » et le processus d'« Arnoldi ». La combinaison de ces trois algorithmes a permis, dans de nombreux cas, d'accélérer la résolution de systèmes linéaires de grande taille. Alors nous disposons ainsi d'une base théorique solide sur laquelle élaborer nos implantations sur la plate-forme de supercalculateur et la plate-forme de grilles XtremWeb.

Dans les chapitres suivants, nous continuerons à montrer en détail les réalisations de notre algorithme sur les machines à plusieurs nœuds à mémoire partagée (Chapitre 4) et sur une grille de calcul légère XtremWeb (Chapitre 6).

Chapitre 4 Algorithmes Parallèles Asynchrones sur Machine à Plusieurs Nœuds à Mémoire Partagée

4.1 Description

Nous allons maintenant adapter la méthode GMRES/LS-Arnoldi asynchrone en PVM, fonctionnant sur plusieurs machines hétérogènes au LIFL, pour la machine IBM SP3 du CRI de l'USTL utilisant MPI.

Cette première mise en œuvre de la méthode hybride GMRES (m1)/LS(k,l)-Arnoldi(m2) décrite dans la section 3.2.3 utilise au maximum les ressources d'une machine parallèle : IBM SP3. Elle permet d'avoir un déroulement totalement asynchrone des différents algorithmes, solution qui, si les paramètres sont bien choisis, évite les délais d'attente et offre des performances optimisées. Les particularités de cet asynchronisme seront discutées dans la section 4.3

4.1.1 Organisation générale

L'idée générale est d'utiliser une machine parallèle, afin que la résolution du système linéaire et le calcul des valeurs propres puissent s'exécuter simultanément de façon indépendante. Les valeurs propres trouvées pourront alors être exploitées de manière asynchrone par l'algorithme itératif de résolution du système pour accélérer sa convergence. Par souci d'efficacité, la partie séquentielle de l'algorithme sera confiée à un seul processeur de la machine.

La réalisation de l'implantation de cette version de la méthode hybride, utilise une seule machine parallèle : Une partie des processeurs de la machine vont exécuter l'algorithme 4 (FIG 3.10) mettant en œuvre la méthode GMRES(***m***)/LS(***k,l***), Le plus grand nombre de processeurs seront donc utilisés pour la méthode GMRES(***m***), car c'est elle qui donnera la solution du système linéaire et que nous voulons accélérer. De plus, comme cela sera montré dans la section 4.4, une vitesse relative trop importante de la méthode d'Arnoldi

détériore la convergence de GMRES(***m***) par une prise en compte trop fréquente de nouveaux vecteurs de redémarrage.

Plusieurs processeurs exécuteront l'algorithme 3 (FIG 3.8) décrit dans la section 3.2.3 qui calculera de manière autonome les valeurs propres de la matrice par la méthode d'Arnoldi. Les différentes parties de cet algorithme sont facilement parallélisables (projection d'Arnoldi, calcul du résidu, redémarrage) à l'exception du calcul des valeurs propres et vecteurs propres par la méthode du QR. Nous utilisons un package spécial de calcul parallèle pour obtenir les valeurs propres utilisant la méthode Arnoldi. Ce package est PARPACK qui est très efficace pour calculer les valeurs propres sur les machines parallèles par la méthode d'Arnoldi Implicite. Il s'agit d'une variante de la méthode d'Arnoldi que nous ne décrivons pas ici. Ses performances en parallèle ne sont pas toujours optimales mais la convergence est en générale bonne. Dans les expérimentations en PVM, nous utilisions notre propre méthode d'Arnoldi mais nous avons ici préféré utiliser un logiciel venant d'une bibliothèque reconnue pour n'avoir à justifier que les apports liés à l'hybridation. Dans un second temps, il serait possible d'utiliser la méthode parallèle asynchrone hybride « Multiple Explicit Restart Arnoldi Method » introduite par Nahid Emad et Guy Edjlali.

Dans la méthode d'Arnoldi, les valeurs propres obtenues au cours d'une itération donnée sont, en général, calculées avec une meilleure précision que lors de l'itération précédente. Seules les valeurs propres dont le résidu est inférieur au seuil de précision choisi seront prise en compte. Le choix de ce seuil ε_a est délicat. Si sa valeur est faible, les valeurs propres seront bien calculées et l'efficacité de la méthode hybride sera accrue. Mais une grande précision peut nécessiter un temps de calcul très important et dans ce cas une longue attente des valeurs propres rendra la méthode inopérante. Les valeurs de ε_a donnant une efficacité satisfaisante dépendent de la matrice, et ont varié dans les tests de 10^{-3} à 10^{-23}.

Il faut aussi choisir le nombre minimum RMIN de valeurs propres, ayant la précision désirée, nécessaire avant l'envoi de ces valeurs pour leur prise en compte dans l'hybridation, et avant le redémarrage de la méthode d'Arnoldi avec un nouveau vecteur initial. Les meilleurs résultats ont été obtenus en choisissant un RMIN petit, et en

accumulant les valeurs propres provenant de plusieurs itérations Arnoldi avec différents vecteurs initiaux, plutôt qu'en choisissant des valeurs plus importantes de RMIN.

Parmi les valeurs propres trouvées, seules celles dont la norme résiduelle sera inférieure à la précision ε_a choisie, seront retenues. Quand leur nombre sera suffisant ($\geq$ RMIN), elles seront envoyées au processus dédié à la partie séquentielle de l'hybridation. Cette partie est exécutée séquentiellement car elle n'utilise qu'un faible volume de données dont la distribution parallèle n'apporterait aucun gain de temps. Il s'agit ici de calculer les paramètres « Least Squares », c'est-à-dire les paramètres du polygone convexe délimitant la zone contenant les valeurs propres dans le plan complexe (voir FIG 3.9), les paramètres de l'ellipse de plus petite aire contenant ce convexe, et les coefficients du polynôme « Least Squares ».

Ce calcul ne va utiliser qu'une partie des valeurs propres, celles trouvées avec une précision suffisante ($< \varepsilon_a$) à l'étape précédente. Le convexe obtenu par ce calcul pourra donc entourer une zone plus petite que le convexe exact. Un résultat satisfaisant ne sera trouvé qu'à partir d'un nombre minimum de valeurs propres transmises.

Ces paramètres seront alors envoyés afin d'exécuter la partie parallèle de l'hybridation. Les processus déroulant l'algorithme GMRES (***m***) vont recevoir ces données de manière asynchrone, à la fin de l'itération en cours. L'algorithme GMRES(***m***) sera alors arrêté, et le dernier itéré GMRES(***m***) ainsi que les paramètres « Least Squares » reçus seront utilisés pour réaliser la partie parallèle de l'hybridation, soit ***l*** itérations « Least Squares ». Ensuite, l'algorithme GMRES (***m***) sera redémarré avec le nouvel itéré x_0 obtenu par la méthode « Least Squares ». Le résidu de ce vecteur sera aussi envoyé à la machine chargée de l'algorithme d'Arnoldi, afin de l'utiliser comme un meilleur vecteur initial pour le prochain redémarrage de cette méthode. Il existe des variantes à cette technique ; par exemple on peut continuer à affiner les valeurs propres sans prendre en compte les calculs réalisés par les autres méthodes.

Principales étapes du calcul

En résume, les principales étapes de la méthode hybride sont présentées sur la FIG 4.1, i.e. :

- le calcul des valeurs propres par la méthode d'Arnoldi parallèle, cette partie utilise le package PARPACK fonctionnant efficacement sur plusieurs processeurs.

- le calcul séquentiel des paramètres « Least Squares »

- la résolution du système linéaire par la méthode GMRES (***m***) parallèle hybridée par la méthode « Least Squares » à chaque réception des paramètres s'y rapportant.

- Le renvoi du résidu de l'itéré de la méthode « Least Squares » au processus PARPACK de la méthode d'Arnoldi pour lui servir de nouveau vecteur initial.

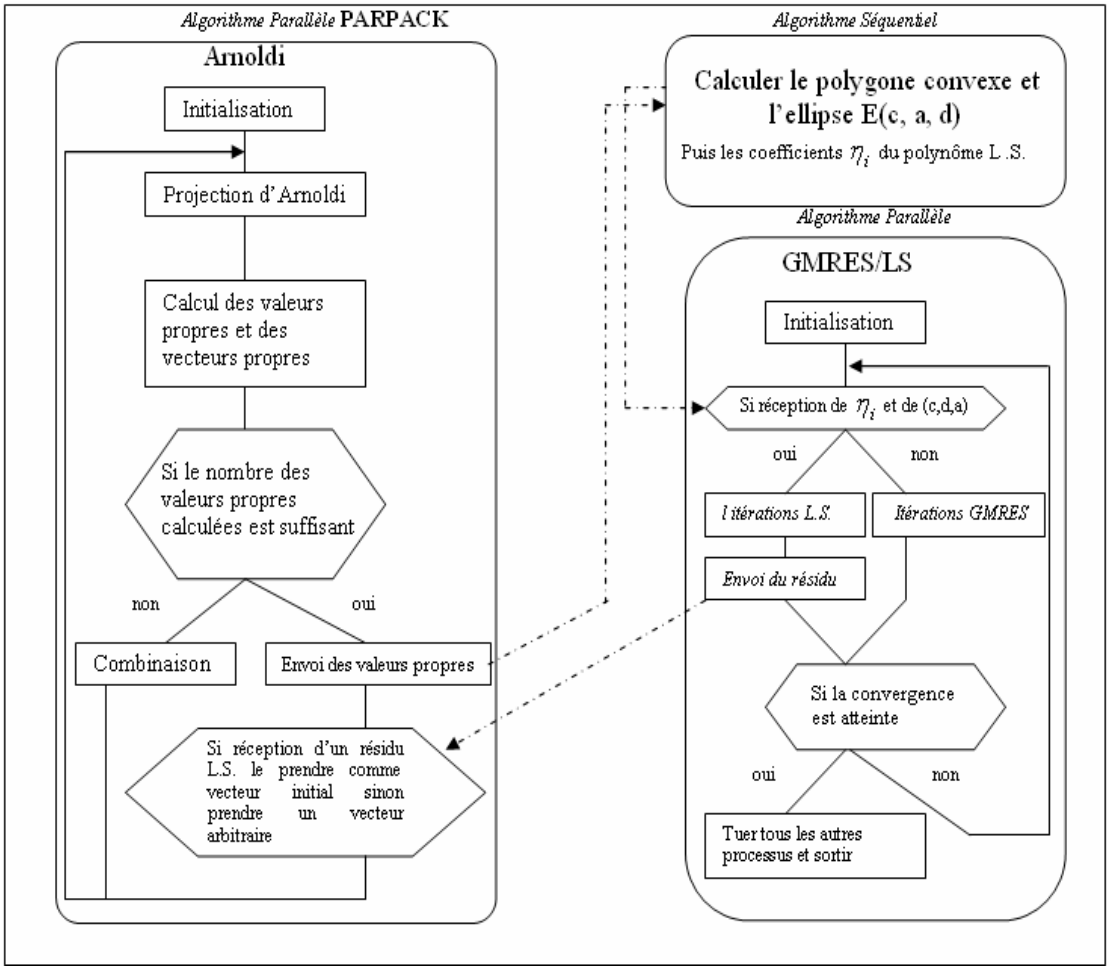


*FIG 4.1*Schéma de principe de la méthode hybride GMRES/Least Squares-Arnoldi parallèle hétérogène asynchrone

4.1.2 Accumulation des valeurs propres

Il est indispensable d'accumuler les valeurs propres calculées pendant plusieurs itérations pour obtenir une meilleure distribution de ces valeurs. En fait, le nombre de valeurs propres utilisables (c'est-à-dire dont la norme résiduelle est inférieure à la précision demandée ε_a), trouvées après chaque itération de la méthode d'Arnoldi n'est pas toujours suffisant pour définir de façon correcte le convexe de la méthode « Least Squares » ainsi que les paramètres associés. En outre, pour un vecteur initial donné, d'une itération sur l'autre les valeurs propres trouvées avec une précision suffisante ne sont pas toujours nécessairement identiques.

D'ailleurs, il est impossible d'obtenir toutes les valeurs propres d'une matrice par la méthode d'Arnoldi. Pour acquérir une bonne répartition des valeurs propres calculées, plusieurs vecteurs initiaux sont employés. On change de vecteur lors des redémarrages successifs de la méthode d'Arnoldi, notamment une fois que l'on reçoit un nouveau

vecteur résidu de l'algorithme GMRES/LS, qui sera alors utilisé pour le prochain redémarrage. De cette manière, à mesure des différentes itérations de la méthode d'Arnoldi, un échantillonnage satisfaisant des valeurs propres sera élaboré, permettant de calculer des paramètres cohérents pour la méthode « Least Squares ».

Pour profiter simultanément des valeurs obtenues après plusieurs redémarrages de la méthode d'Arnoldi, il est important de mémoriser les valeurs trouvées au cours des itérations. Cette accumulation est nécessaire afin d'obtenir un nombre suffisant de ces valeurs, avec une distribution représentative du convexe les contenant toutes. Il sera alors possible de les utiliser efficacement pour élaborer les paramètres « Least Squares ». Toutes ces valeurs mémorisées seront mises à contribution à chaque mise à jour de ces paramètres. Il pourra arriver que des itérations successives de l'algorithme d'Arnoldi redonnent une valeur propre déjà calculée, qui sera alors dupliquée. Mais ces copies ne modifieront pas les caractéristiques du polygone convexe les contenant (voir FIG 3.9)

4.2 Choix d'implantation

4.2.1 Répartitions des processus

La méthode hybride, implantée sous sa forme asynchrone, va donc être gérée par différents processus s'exécutant simultanément sur différents processeurs de la machine-- IBM SP3 (voir FIG 4.2)

La plupart des processeurs auront en charge l'algorithme 4 (GMRES (m)/LS(k,l)) sous forme SPMD ; avec un processus gestionnaire et ***p*** processus calculant identiques. Les processeurs calculants lisent leurs propres données directement et exécutent la partie de la méthode GMRES(***m***) correspondante, communicant avec leurs processus frères. Les lignes des matrices testées ayant en moyenne le même nombre d'éléments non nuls, la charge se trouve équilibrée. La partie séquentielle de cet algorithme (basée sur la factorisation QR) sera faite de manière redondante sur chaque processeur. Une parallélisation de cet algorithme serait possible, mais n'est pas justifiée ici car les calculs de cette section ont une complexité en $O(m^3)$, avec m petit devant n ($m<<n$). Chaque processeur calculant étant aussi performant qu'une machine séquentielle rapide, cette

redondance n'est pas pénalisante par rapport à un transfert sur une station séquentielle et elle évite même des communications coûteuses.

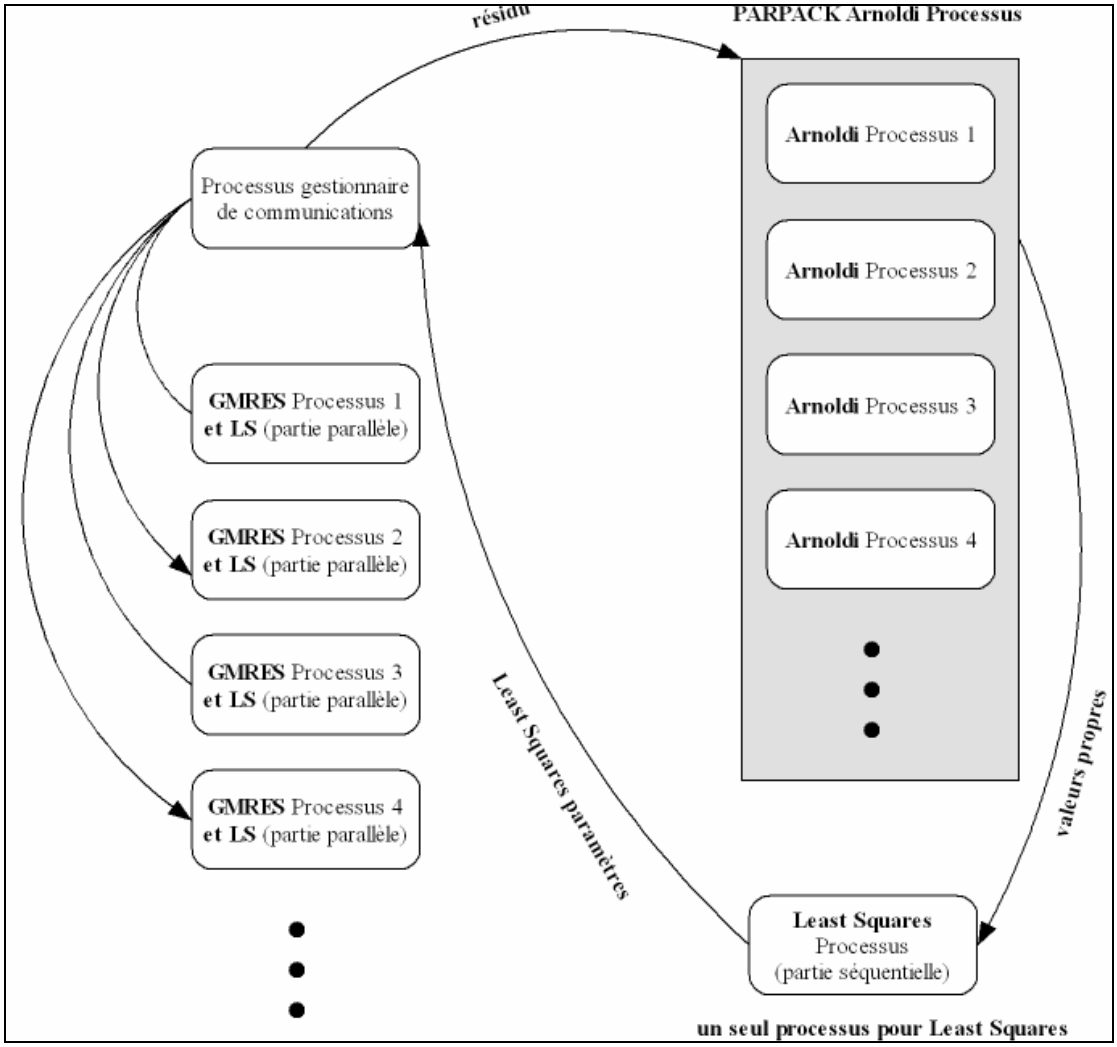


FIG 4.2–Schéma général de l'implantation parallèle hétérogène asynchrone de la méthode hybride GMRES/Least Squares-Arnoldi

Les processeurs consacrés aux calculs liés à PARPACK comprennent la réception du résidu, la projection d'Arnoldi et le calcul des valeurs propres.

Un seul processeur calcule les paramètres « Least Squares » (les sommets du polygone convexe, les paramètres de l'ellipse et les coefficients du polynôme « Least Square »), qui seront ensuite envoyé aux processeurs exécutant l'algorithme GMRES(***m***). Les avantages de l'existence de ce processus intermédiaire sont les suivants :

- ✓ que le processus d'Arnoldi ne connaisse qu'un seul interlocuteur.

- ✓ que les données ne transitent qu'une fois sur le réseau.
- ✓ que la réception de ces données par les processeurs frères SPMD exécutant chacun l'algorithme GMRES/LS soit synchronisée, afin que la prise en compte intervienne après le même nombre d'itérations GMRES(***m***) pour chaque processus et que la structure de données distribuée reste cohérente.

A la fin de chaque itération GMRES(***m***), les processus regarderont si des paramètres « Least Square » sont arrivés. Dans ce cas, ceux-ci seront pris en compte pour exécuter *l* itération de la méthode « Least Squares » avant de faire un redémarrage de GMRES(***m***). Le résidu du vecteur itéré obtenu à ce moment sera envoyé au processus gestionnaire de communications qui le transmettra aux processeurs affectés au package « PARPACK » pour obtenir un meilleur vecteur initial lors du prochain redémarrage de l'algorithme d'Arnoldi. Les principaux paramètres d'implantation de cet ensemble de processus sont décrits par FIG 4.3. L'influence des plus importants d'entre eux est discutée dans les sections suivantes.

Paramètres liés aux méthodes numériques :

n	taille de la matrice
m(GMRES)	taille de l'espace de projection pour la méthode GMRES(***m***)
m(Arnoldi)	taille de l'espace de projection pour la méthode d'Arnoldi
k	degré du polynôme « Least Square »
l	nombre d'itérations « Least Square » après chaque prise en compte des paramètres calculés pour cette méthode.

Paramètres liés à la mise en œuvre :

c	largeur des tableaux ELLPACK.
p	nombre de processeurs engagés pour exécuter les processus GMRES/LS
ε_g	valeur limite de la norme résiduelle pour la convergence de la méthode GMRES(***m***)
ε_a	valeur limite de la norme résiduelle pour le calcul des valeurs propres par la méthode d'Arnoldi
ITERMIN	nombre minimum d'itérations GMRES(***m***) entre deux prises en compte de paramètres « Least Square » (afin d'éviter une divergence : voir la section 4.4)
RMIN	nombre minimum de valeurs propres à trouver par la méthode d'Arnoldi avant de changer de vecteur initial
NVPMIN	nombre minimum de valeurs propres à accumuler avant de calculer les paramètres « Least Squares »

FIG 4.3–Principaux paramètres de l'implantation parallèle hétérogène asynchrone de la méthode hybride GMRES (m)/LS-Arnoldi

4.2.2 Mise en place de la matrice

Le stockage de matrices de grande taille pose de sérieux problèmes d'occupation des disques. Une matrice est lue simultanément sur les deux groupes de processeurs (le groupe de GMRES(***m***) et le groupe de PARPACK) afin de limiter les échanges de données par le réseau. Cette duplication des données, coûteuse en espace mémoire, présente l'avantage de rendre complètement indépendantes les deux parties de l'application, et de permettre un fonctionnement totalement asynchrone de l'ensemble. On utilisera la technique suivante : chaque processeur lit uniquement le morceau de la matrice dont il a besoin. Cette méthode nous permet d'économiser beaucoup d'espace mémoire pour les très grandes matrices.

Comme il s'agit de matrices creuses, les éléments nuls ne sont pas stockés et le format de compression que nous avons utilisé est le CSR (voir FIG 3.1) pour diminuer la taille des messages lors de la distribution des données.

4.3 Asynchronisme

4.3.1 Indéterminisme

Sauf lorsque la SP3 fonctionne en mode « batch » où un processus fonctionnant sur un processeur occupe le processeur entier et ne partage pas de temps avec les autres processus, les machines qui font tourner les différents processus de la méthode hybride travaillent en temps partagé avec d'autres utilisateurs. Par conséquent la charge de travail de chacune d'entre elle évolue indépendamment de celle des autres, et de manière imprévisible au cours du temps. Pour une exécution donnée, le calcul hybride va coupler les valeurs propres de l'itération n°P1 de l'algorithme d'Arnoldi implanté sur les processeurs de « PARPACK », avec l'itéré n°G1 de l'algorithme GMRES(***m***) implanté sur les processeurs de GMRES(***m***). Si nous refaisons l'exécution du programme avec la même matrice et sans avoir modifié aucun paramètre, les valeurs propres issues de la même itération Arnoldi n°P1 peuvent être couplées cette fois avec l'itéré de GMRES(***m***) n°G2 ($\neq$ G1), du fait que les charges de travail et donc les vitesses des différents processeurs n'auront pas évolué de la même façon (voir FIG 4.4). Il en résulte un non déterminisme du calcul hybride et une non reproductibilité des valeurs de contrôles affichées (voir *FIG 4.5*). Évidemment, ceci influence le nombre d'itérations pour atteindre la convergence souhaitée mais, a priori, cela est sans incidence sur le résultat numérique

final. L'accélération de la convergence grâce à l'utilisation de la présente méthode hybride peut ainsi se trouver affectée par cet indéterminisme.

Afin d'estimer les performances de la méthode hybride, il est préférable d'éviter cet indéterminisme. C'est pourquoi nous avons fait nos tests en mode « batch », par soumission de nos tâches à la file d'attente de ce mode.

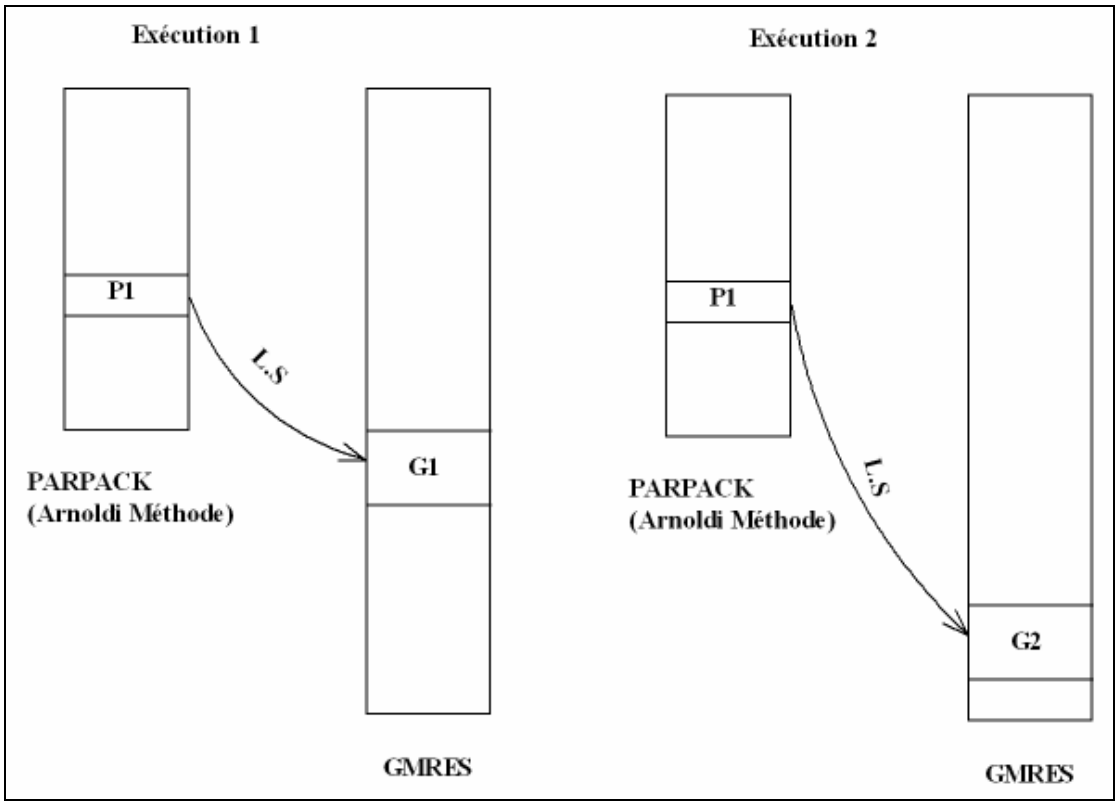


FIG 4.4–Asynchronisme des processus pour la prise en compte des valeurs propres dans la méthode hybride asynchrone parallèle.

4.3.2 Précautions nécessaires pour l'asynchronisme

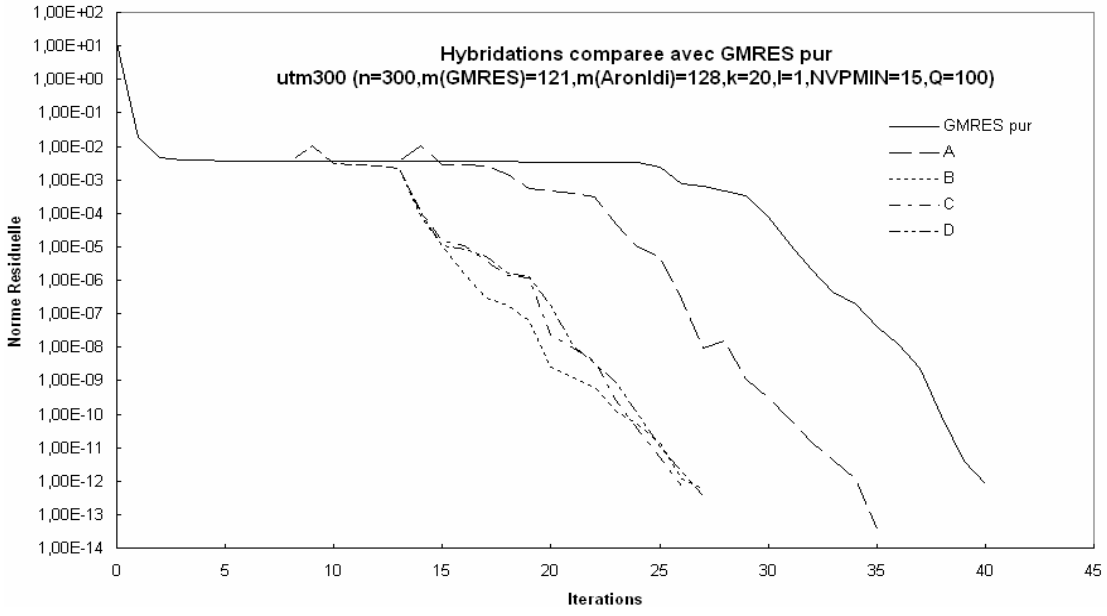
La réception des données est asynchrone. il peut arriver que les valeurs de l'itération n°i arrivent avant celles de l'itération n°i-1. Il est obligatoire d'avoir un estampillage des messages pour éviter toute confusion (surtout au niveau des échanges entre processus GMRES(***m***) au cours des itérations « Least Squares », ainsi que pour la réception des paramètres calculés à partir des valeurs propres.

Il faut mettre en place une barrière de synchronisation pour la prise en compte des paramètres de la méthode « Least Squares » par les processus GMRES(***m***) exécutant les itérations GMRES/LS, pour que cette prise en compte, quand elle a lieu, soit validée par

tous les processus frères au même numéro d'itération. En fait, malgré la présence de plusieurs synchronisations par itération de la méthode GMRES(***m***) dues aux communications bloquantes (échanges des parties de $\boldsymbol{v}$, $\boldsymbol{h^j}$, $\boldsymbol{x}$ détenues par chaque processeur aux étapes n°2 et 3 de l'algorithme 2, voir FIG 3.6), la réception des paramètres de la méthode « Least Squares » peut se trouver décalée d'un processus sur l'autre, soit par une vitesse d'exécution différente des processus, soit par une durée d'acheminement différente du message. Il est suffisant que le message arrive avant l'instruction de réception pour un processus, et après pour un autre, pour créer une distorsion. Les processus qui ont reçu les paramètres vont alors se lancer dans une itération « Least Squares » tandis que les autres feront une itération GMRES(***m***). Certainement, le prochain échange de message se terminera par un blocage (« dead-lock »), les identifiants (« tags ») des messages émis et attendus étant incompatibles. D'où la nécessité de mettre en place un mécanisme pour synchroniser la réception de ces paramètres.

4.3.3 Convergence et Synchronisation

L'accélération de la convergence, visée par cette méthode parce que la diminution du nombre d'itérations pourra réduire la durée du calcul d'une façon significative, est sujette à la répartition des valeurs propres trouvées par la méthode d'Arnoldi, et aux résultats approchés des itérations GMRES(***m***) précédentes. Choisir le meilleur moment pour le couplage afin que l'accélération soit optimale est un problème ouvert. Pourtant certains paramètres ont une influence capitale, cf. les paragraphes suivants.



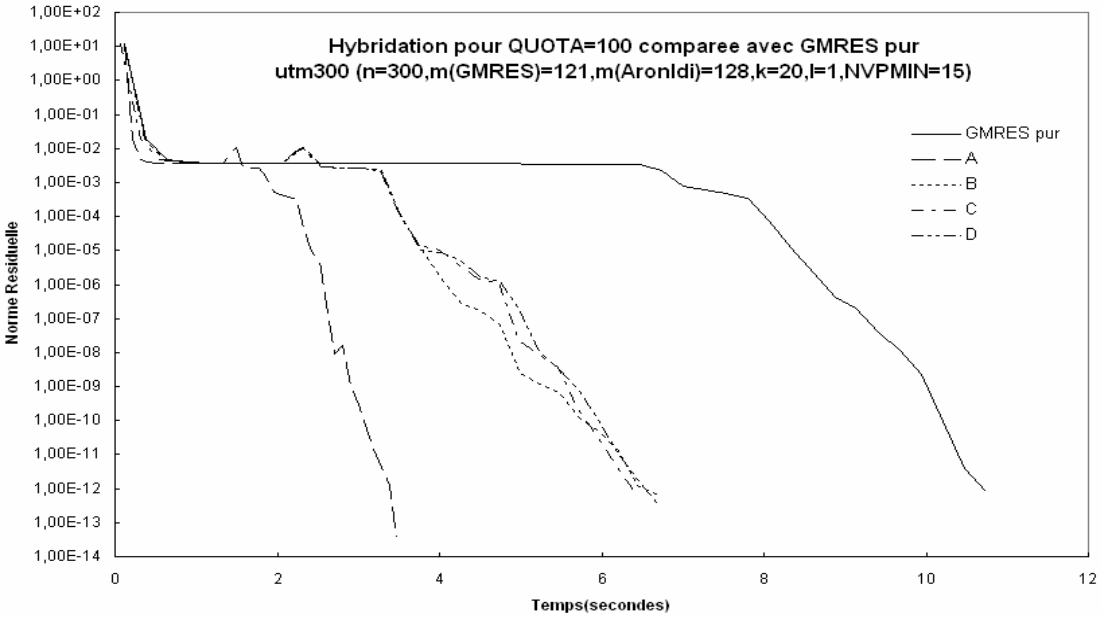


FIG 4.5 –Non déterminisme de l'accélération de la convergence (matrice « utm300 »). Variation du résidu pour quatre exécutions A, B, C et D avec les mêmes paramètres.

Précision pour le calcul des valeurs propres

La précision désirée pour l'obtention des valeurs propres influe non seulement sur la durée du calcul ε_a des valeurs propres mais aussi sur l'efficacité de l'hybridation « Least Squares ». En effet, si la précision demandée est plus grande, il faudra plus d'itération à l'algorithme d'Arnoldi pour calculer un nombre suffisant de valeurs propres pour obtenir des paramètres « Least Squares » significatifs, ce qui induira une ε_a première prise en compte plus tardive de ces paramètres, puis une fréquence des itérations « Least Squares » plus réduite. Si au contraire correspond seulement à une précision médiocre, on aura rapidement beaucoup de valeurs, mais celles-ci seront moins bien calculées, alors les paramètres déduits de « Least Squares » seront moins efficaces.

Par conséquent, cette précision influe sur le moment du couplage ainsi que sur sa fréquence. Pour une matrice donnée, il n'est pas certain que l'accélération la plus rapide de la convergence soit obtenue avec l'exigence la plus grande précision des valeurs propres. On peut aussi observer d'après les tests que l'efficacité de la méthode hybride est meilleure lorsque la première itération « Least Squares » intervient assez tôt. Les matrices testées ont montré de très grandes différence de l'une à l'autre quant à la valeurs de ε_α donnant les meilleures performances (de 10^3 à 10^{-12}).

Taille des espaces de projection

Les tailles ***m***(GMRES) et ***m***(Arnoldi) choisies pour les espaces de projection des méthodes GMRES(***m***) et Arnoldi ont également une grande influence sur le couplage, en agissant elles aussi sur l'efficacité de la prise en compte des valeurs propres ainsi que sur les temps de calculs. En fait, le volume des données à calculer, et donc la durée d'une itération dépendent de la taille du sous-espace pour chacune de ces deux méthodes. La convergence de la méthode hybride, influencée par les moments des couplages correspondant aux prises en compte des valeurs propres par la méthode « Least Squares », dépendra pareillement des valeurs relatives de ces deux tailles.

Si ***m***(Arnoldi) est trop petit, le calcul des valeurs propres sera rapide mais peu de valeurs propres sera obtenues , puis le convexe calculé par la méthode « Least Squares » sera très éloigné du convexe idéal qui contient toutes les valeurs propres. Dans ce cas les prises en compte des paramètres « Least Squares » sont rapprochées mais peu efficaces. Au contraire, si ***m***(Arnoldi) est grand, beaucoup de valeurs sont obtenues et le convexe calculé se rapprochera bien du convexe idéal. Mais le calcul des valeurs propres, qui recourt à une matrice de Hessenberg de taille ***m***(Arnoldi), sera alors lent parce qu'il devra calculer un volume de données important. Pour ***m***(Arnoldi) élevé, les itérations « Least Squares », conséquences du couplage entre les composants de la méthode hybride, seront donc espacées. Normalement, si elles le sont trop, l'effet réduit sur la convergence sera nul. En poussant chacun de ces deux cas à l'extrême, l'influence de l'hybridation en sera réduit (soit en raison d'une fréquence de couplage insuffisante, soit à cause d'un couplage fréquent mais peu efficace), par conséquent les performances se rapprocheront de celles de la méthode GMRES(***m***) pure (voir FIG 4.13).

Par conséquent, le choix des valeurs de ***m***(Arnoldi) et de ***m***(GMRES) permettant d'acquérir une efficacité optimale est délicat. Mais ce choix n'est pas simple, car il dépend beaucoup de la structure et les données de la matrice utilisée, et il n'est pas facile, a priori, de prédire les meilleurs paramètres. De façon empirique les valeurs de ***m***(GMRES) et ***m***(Arnoldi) appliquées au cours des tests décrits dans la section 4.4 ont été déterminées en s'inspirant les valeurs donnant les meilleures performances au cours de tests préliminaires.

4.4 Résultats et Analyses

Nous avons comparé systématiquement les performances relevées sur les tests avec la méthode hybride avec celles de la méthode GMRES(***m***) pure, dans les mêmes conditions de parallélisme. Une répartition sur huit processeurs du calcul de la méthode hybride où GMRES(***m***)/LS-Arnoldi a été systématiquement employée (5 processeurs pour GMRES(***m***), 2 processeurs pour Arnoldi, 1 processeurs pour LS).Aucun préconditionnement n'a été utilisé, car il aurait été impossible d'employer le même, avec la même efficacité, sur des matrices différentes, et il se révèle plus rigoureux d'employer toujours la méthode de référence. D'ailleurs, en ajoutant des paramètres délicats à optimiser, mettre en œuvre un préconditionnement peut compliquer beaucoup le problème, d'autant plus que ce préconditionnement devrait être établi par un calcul supplémentaire qui précède l'ensemble des calculs ici décrits. Il est évident que la méthode hybride, qui est plus complexe à mettre en place, s'appliquera particulièrement bien aux cas où les convergences sont difficiles à atteindre. Et pour ces cas là, on ne connaît non plus les préconditionnements vraiment efficaces.

Seuls les nombres d'itération de la méthode GMRES(***m***) sont présentés, nous ne donnons par le nombre de fois que les méthodes « least square » sont lancées, mais ce nombre peut être observé car chaque itération « Least Square » correspond à un pic de résidu. Ce nombre apparaît donc en comptant les pics du résidu dans les courbes. Par ailleurs, nous donnons le temps de calcul global pour trouver la solution du système.

4.4.1 Matrices choisies pour les tests

Nous avons testé plusieurs systèmes linéaires afin qu'on puisse comparer l'algorithme hybride avec la méthode GMRES(***m***) pure. Afin de pouvoir vérifier l'exactitude des valeurs numériques des solutions trouvées par l'algorithme, le vecteur second membre *b* a été bien choisi tel que la solution du système soit x= (1,1, ... ,1) ^{T} (on a alors $b = Ax = \sum_{i=1}^{n} A_i$). Les méthodes démarrent avec $\boldsymbol{x_0}$= (0,0, ... ,0)^{T} sauf dans quelque cas des tests avec des vecteurs initiaux $\boldsymbol{x_0}$ différents .

Première matrice (« utm1700a ») : Des matrices industrielles, issues du site Web « MatrixMarket » ont été utilisées. La matrice « utm1700a » vient d'une application de physique nucléaire (taille 1700×1700, 21313 éléments non nuls) Les tests effectués ont

dévoilé une convergence difficile de GMRES(***m***) pur avec cette matrice, quand la taille ***m*** est modeste (voir FIG 4.9). D'où l'utilisation de ***m***=400 dans les autres courbes.

Deuxième matrice (« utm300 ») : Venant toujours du site Web « MatrixMarket », la matrice « utm300 » vient également de physique nucléaire (taille 300×300, 18000 éléments non nuls). Les tests effectués ont montré une convergence difficile de GMRES(***m***) pur avec cette matrice, lorsque la taille ***m*** est modeste (voir FIG 4.10).

Troisième exemple (« ck104 ») : Toujours issue du site Web « MatrixMarket », la matrice « ck104» (taille 104×104, 3000 éléments non nuls) a été employée. Les tests effectués ont montré une convergence difficile de GMRES(***m***) pur avec cette matrice, lorsque la taille ***m*** est modeste (voir FIG 4.8).

Quatrième matrice (« utm30600») : Des matrices industrielles, issues du site Web « MatrixMarket » ont été utilisées. Elle vient d'une application de physique nucléaire (taille 3060×3060, 42211 éléments non nuls)

4.4.2 Accélération de la convergence

Les résultats expérimentaux montrent que l'accélération de la convergence est significative, mais dépend beaucoup de la matrice utilisée (voir FIG 4.6,FIG 4.7,FIG 4.8), et d'un choix judicieux des paramètres. En particulier, les paramètres ***k*** (degré du polynôme « Least Squares ») et ***l*** (nombre d'itérations « Least Squares » à chaque couplage hybride) quantifient l'interaction des méthodes numériques de base au sein de la méthode hybride. On observe que le temps de traitement global de la méthode hybride asynchrone parallèle est pratiquement toujours meilleur, et souvent de loin, que celui de la méthode GMRES(***m***) pure parallèle. L'accélération peut même être particulièrement spectaculaire quand la convergence de la méthode GMRES (***m***) pure est difficile (voir FIG 4.10).

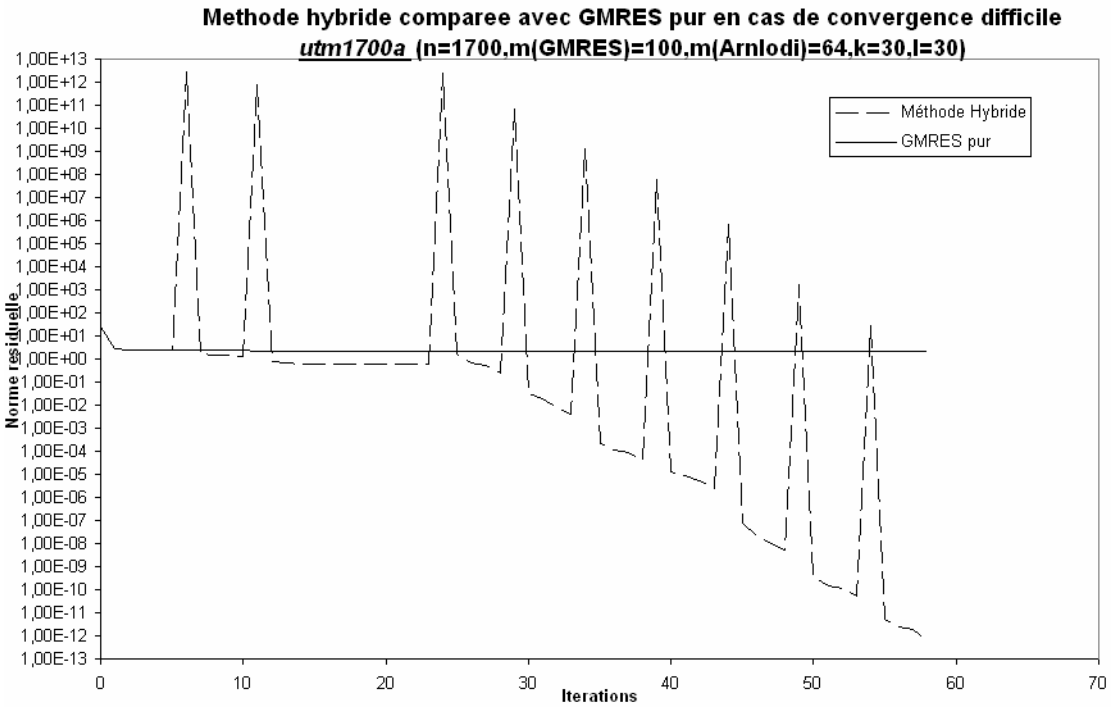
Cette étude expérimentale dévoile plusieurs points de vue qui méritent d'être soulignés. Quand survient la prise en compte des paramètres de l'hybridation par les processus GMRES(***m***), nous remarquons fréquemment une augmentation temporaire, parfois très importante, de la norme résiduelle. Néanmoins les réductions de cette norme résiduelle en découlant sont bien plus rapides qu'avant la prise en compte. Toutefois, malgré ces

pics, la convergence s'accélère à la fin. Les aspects numériques ont développés par Azeddine Essai dans sa thèse [31]. Ce phénomène est également bien expliqué par sa théorie mathématique.

A cause de ces pics, de trop fréquents envois de paramètres « Least Squares » sont nuisibles si nous empêchons chaque prise en compte d'avoir des répercussions sur un nombre suffisant d'itérations GMRES(***m***). Si les pics sont adjacents et rapprochés, nous pourrions même avoir une divergence. Par conséquent, les processeurs exécutant la méthode d'Arnoldi pour le calcul des valeurs propres pourraient être peu nombreux, puisque les paramètres « Least Squares », calculés à partir des valeurs propres trouvées par la méthode d'Arnoldi, ne doivent pas être acquis avec une trop grande fréquence. Le recours au parallélisme est néanmoins nécessaire quand la matrice calculée est de taille importante.

4.4.3 Paramètres importants d'hybridation

Il est possible d'améliorer l'efficacité de l'hybridation « Least Squares » par l'utilisation d'une valeur élevée de la puissance ***l***. Elle désigne le nombre d'itérations « Least Squares » réalisées à chaque prise en compte. Le développement de la norme résiduelle présente alors des pics plus élevés, mais la convergence globale est encore plus rapide. Pourtant, le temps de calcul de la partie parallèle de l'hybridation augmente avec cette puissance ***l***.



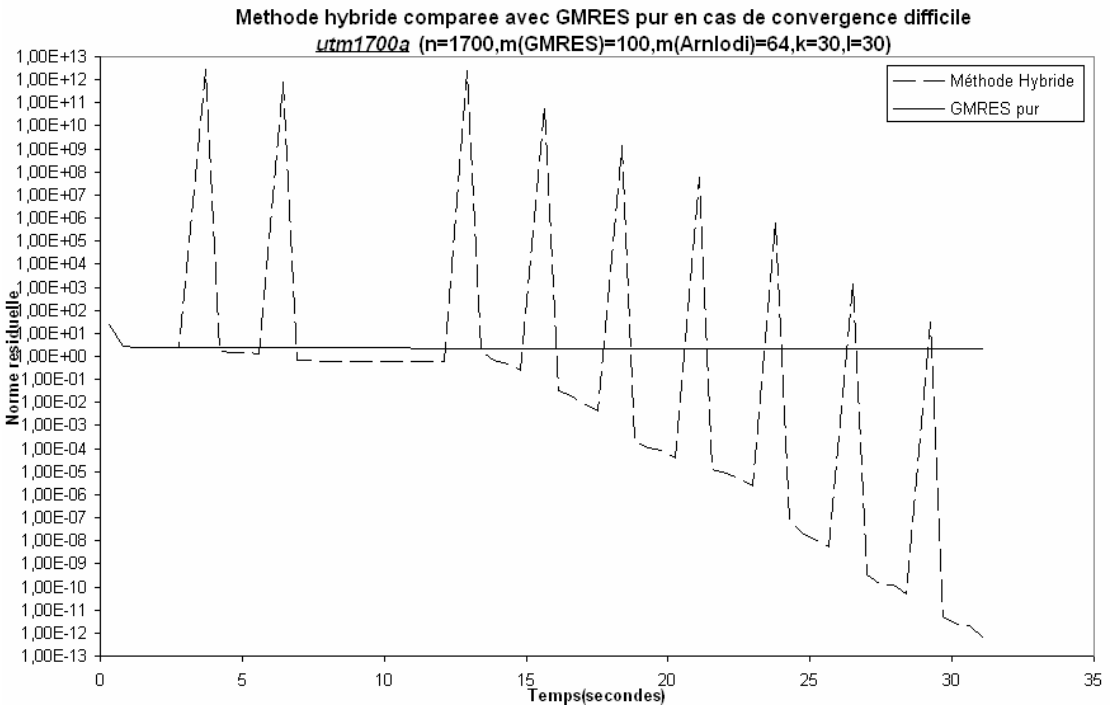
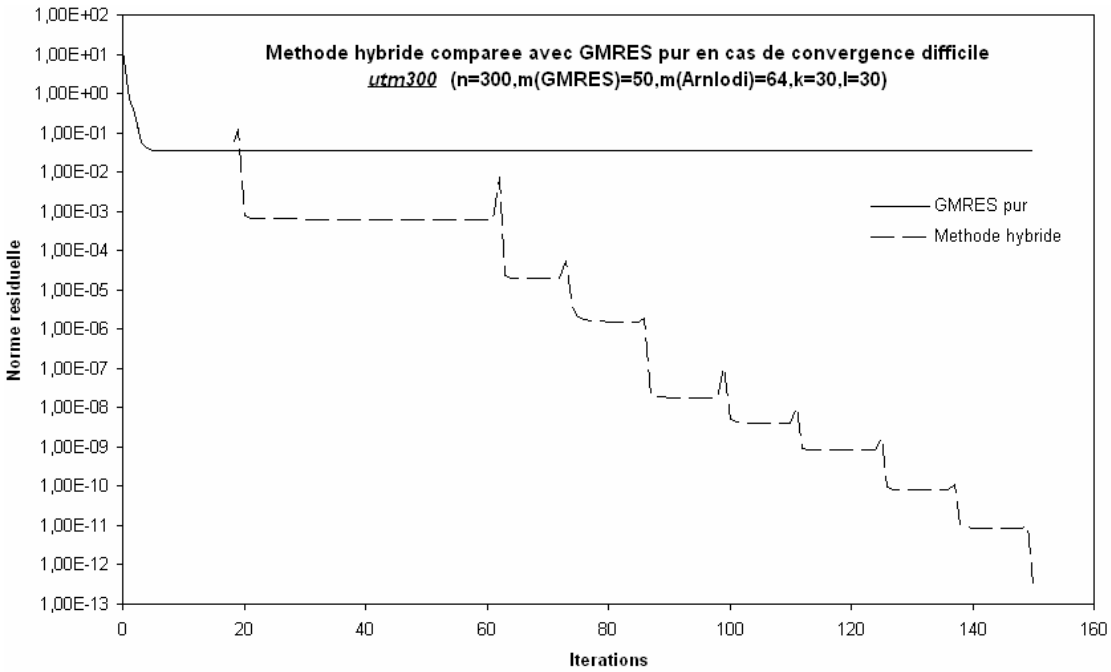


FIG 4.6 -Evolution de la norme résiduelle avec la méthode hybride asynchrone comparée avec la méthode GMRES (m) pure, en cas de convergence difficile (matrice « utm1700a »)



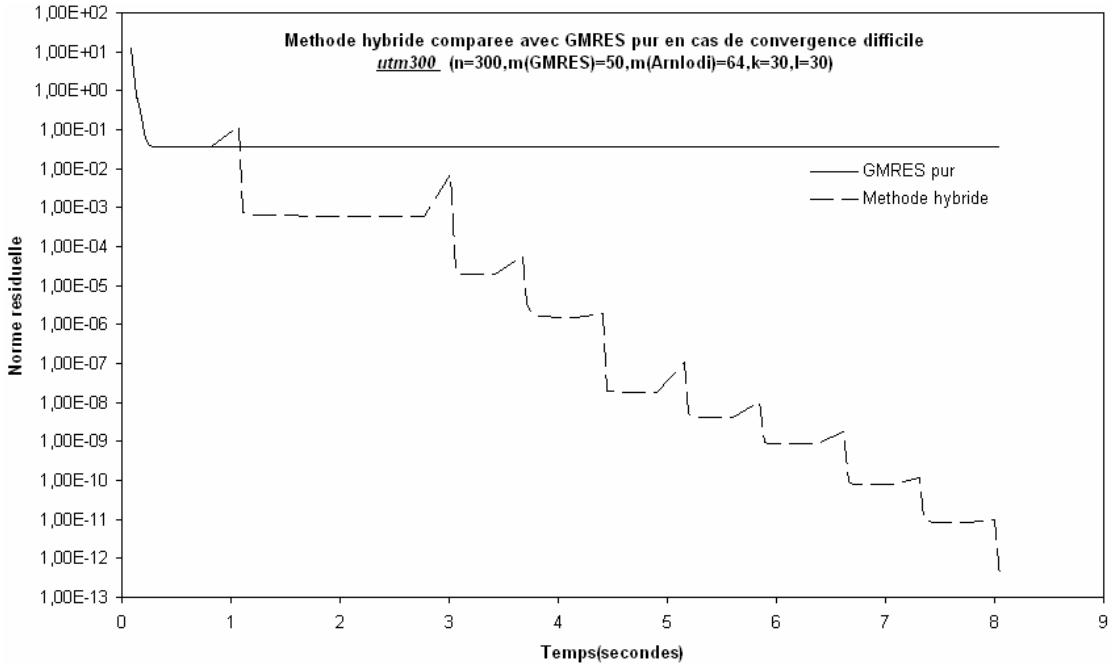
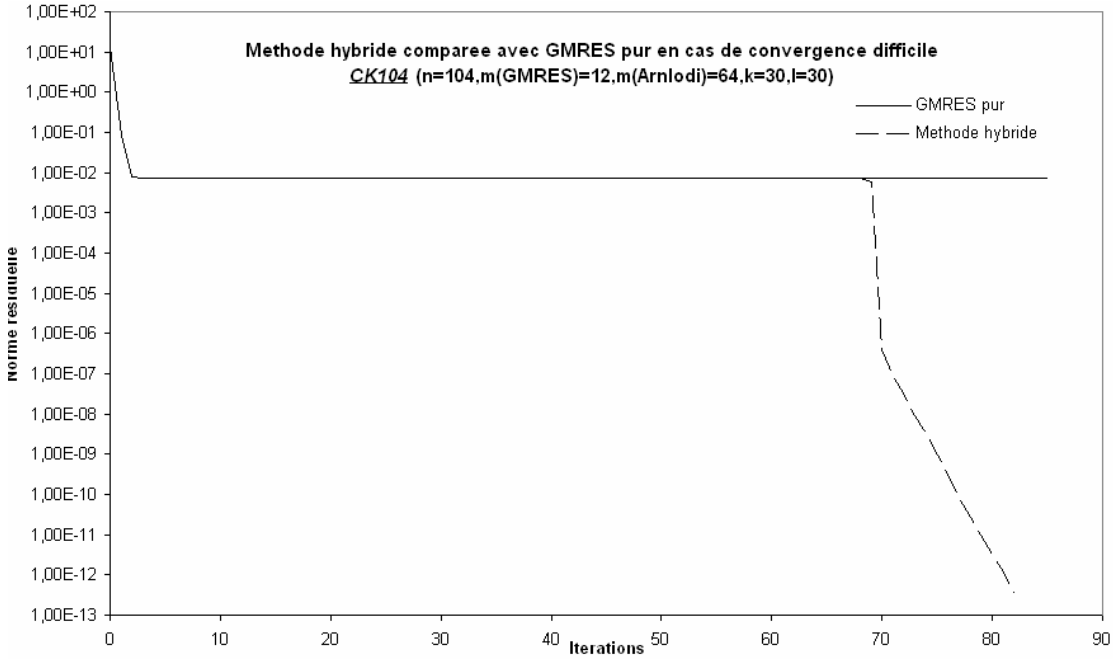


FIG 4.7 –Evolution de la norme résiduelle avec la méthode hybride asynchrone comparée avec la méthode GMRES (m) pure, en cas de convergence difficile (matrice « utm300 »)

Dans cette figure FIG 4.7, nous remarquons une accélération significative de la convergence après l'utilisation de la méthode hybride en cas de convergence difficile pour la méthode traditionnelle GMRES(***m***) pour la matrice « *utm300* ». La forme de la courbe de la convergence de la méthode hybride est presque un escalier qui est convergente à la fin. Dans la figure FIG 4.6, les pics, extrêmement saillants déforment cette sorte d'escalier mais ne nuisent pas à la convergence.



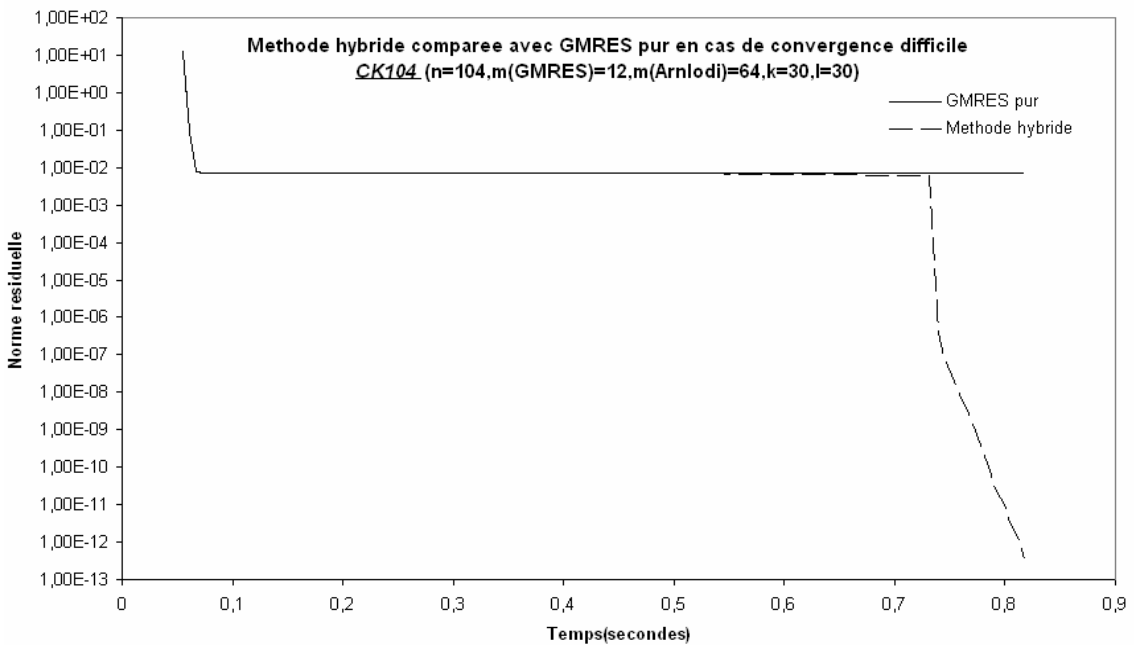
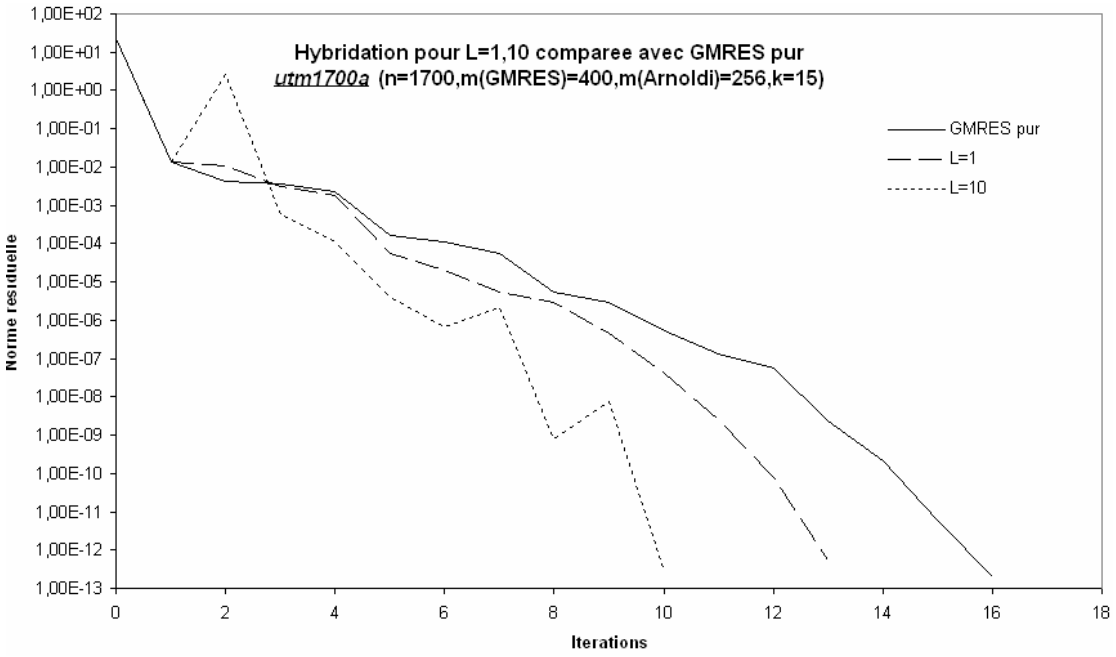


FIG 4.8–Evolution de la norme résiduelle avec la méthode hybride asynchrone comparée avec la méthode GMRES (m) pure, en cas de convergence difficile (matrice « ck104 »)

Dans cette figure FIG 4.8, nous observons une chute rapide après la prise en compte des paramètres de « Least Squares » lors de l'utilisation de la méthode hybride en cas de convergence difficile par la méthode traditionnelle GMRES(***m***) pour la matrice « *ck104* ». La forme de la courbe de la convergence de la méthode hybride est presque une ligne horizontale qui plonge brusquement vers la convergence.



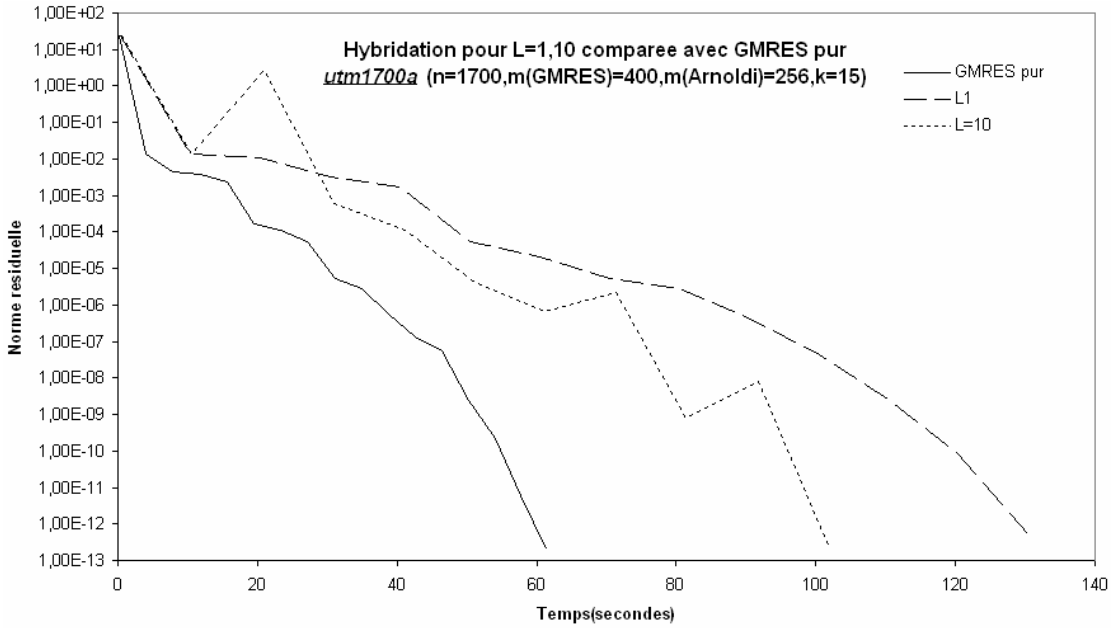
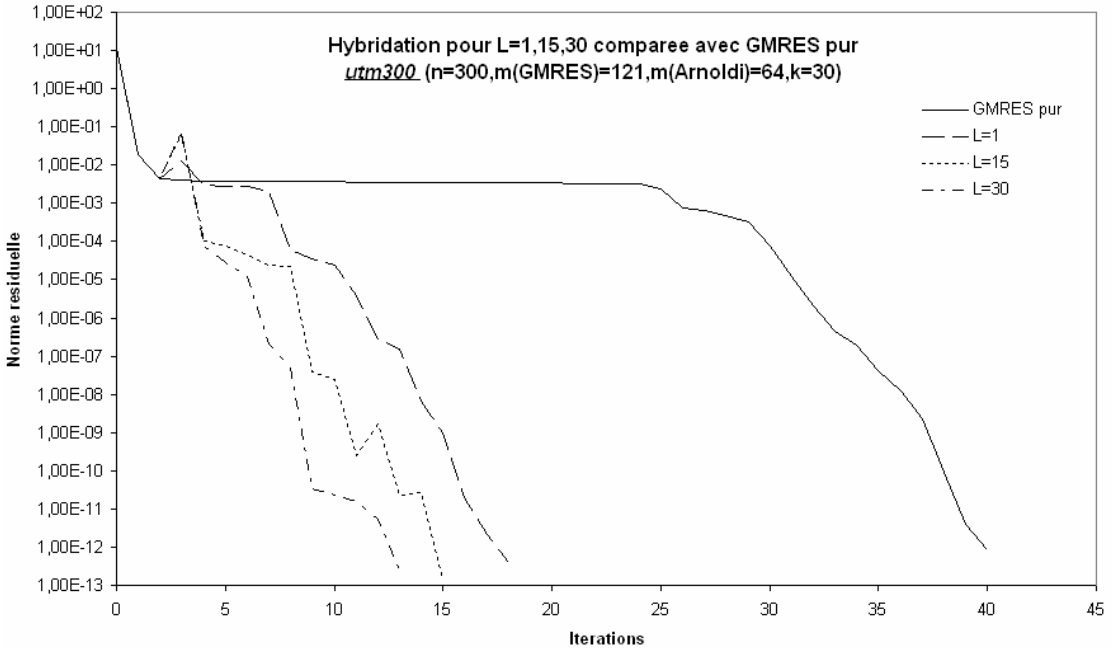


FIG 4.9 -Evolution de la norme résiduelle avec la méthode hybride asynchrone comparée avec la méthode GMRES (m) pure (matrice utm1700a)

Dans cette figure FIG 4.9, le temps nécessaire pour atteindre la convergence de la méthode hybride (***l***=1,10) est plus long que par la méthode traditionnelle GMRES(***m***) pour la matrice « *utm1700a* ». Ici, avec ***m***=400, la méthode GMRES(***m***) présente une efficacité telle que l'hybridation semble superflue, même si le nombre d'itérations se trouve tout de même raccourci, mais le surcoût du calcul « Least Squares » peut être pénalisant, surtout lorsqu'une valeur faible du paramètre ***l*** en limite l'efficacité.



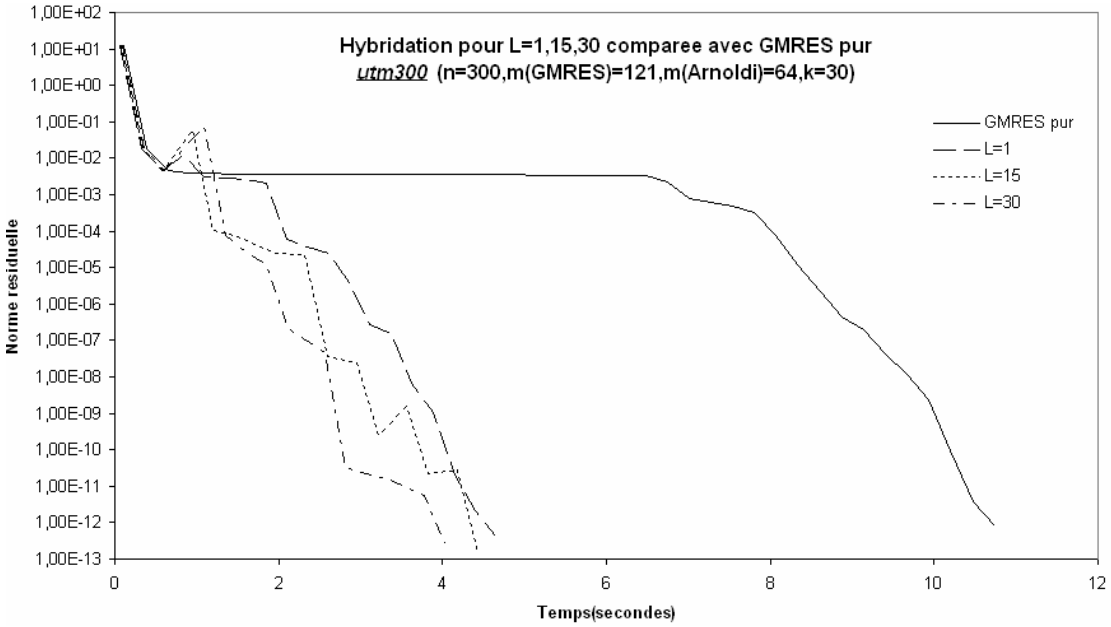


FIG 4.10 -L'influence de L (matrice utm300)

Les effets du surcoût de calcul « Least Squares » sont bien visibles sur le graphe où l'on peut remarquer que la largeur des pics est plus importante sur la courbe des temps que sur celle des itérations (voir FIG 4.10 où les échelles horizontales sont en concordance pour GMRES(***m***) pur). Pour les valeurs de ***l*** petites, l'augmentation du temps de calcul est moindre que le temps économisé avec l'accélération de la convergence. Cependant, pour ***l***=1,10 le surcoût de l'hybridation peut produire un temps plus mauvais que pour GMRES(***m***) (voir FIG 4.9). Mais une augmentation excessive de ***l*** n'arrive pas à économiser de temps supplémentaire, voire cause une perte de temps, lorsque les pics sont trop élevés et que la réduction de la norme résiduelle après de tels pics peut coûter plusieurs itérations. Avec des pics trop importants, les paramètres « Least Squares » suivants peuvent arriver avant la décroissance attendue de la norme résiduelle pour l'itération « Least Squares » précédente et une divergence éventuelle peut alors se produire.

Le degré ***k*** du polynôme « Least Squares » est aussi un paramètre important, et l'augmentation de sa valeur peut également accroître l'efficacité de l'hybridation. Mais, comme pour tous les paramètres précédents, l'augmentation de ***k*** prolonge la durée du calcul parallèle, et les mêmes phénomènes peuvent se produire (voir *FIG 4.12*). Les expériences prouvent qu'en augmentant soit ***l***, soit ***k***, le nombre d'itérations réduit d'abord, puis stagne ou même s'accroît, et on peut trouver des valeurs optimales de ***k*** et ***l***. Ces valeurs apparaissent aussi à propos du temps de calcul.

Le paramètre QUOTA, qui représente le nombre de valeurs propres produits par PARPACK, est également un paramètre très important. Grâce à l'efficacité du logiciel, quand on augmente la valeur de QUOTA de quelques unités, le temps de calcul est quasiment identique. Mais plus nous demandons de valeurs propres et plus le logiciel va adapter ses stratégies de redémarrage pour converger vers un plus grand nombre de valeurs propres. Néanmoins, l'objectif du logiciel est alors de converger vers QUOTA valeurs propres et les premières valeurs propres vont converger peut-être plus doucement avec la taille de sous-espace MA fixée. Par ailleurs en demandant plus de valeurs propres au logiciel nous pouvons espérer trouver des valeurs propres différentes à chaque calcul, pour une précision donnée, et donc avoir un plus grand nombre de valeurs propres distinctes pour construire une meilleure ellipse pour la méthode des moindres carrés. Pour toutes ces raisons, nous observons bien, là aussi, que le choix de la valeur optimale est délicat (voir FIG 4.16,FIG 4.17).

Le vecteur initial de PARPACK a aussi une influence. On combine en général les résidus qu'on a obtenu de GMRES (m) pour fabriquer le vecteur de redémarrage. L'influence du choix de redémarrage dépend des caractéristiques des matrices. Chaque matrice a son type de redémarrage le plus efficace.
Pour les vecteurs initiaux de PARPACK:

$$x = \alpha \times V$$

x est le vecteur initial, est un α coefficient d'un vecteur, V est une matrice de deux dimension contenant les valeurs propres.

Type 1, α est un vecteur plein de 1 avec la taille « QUOTA ».

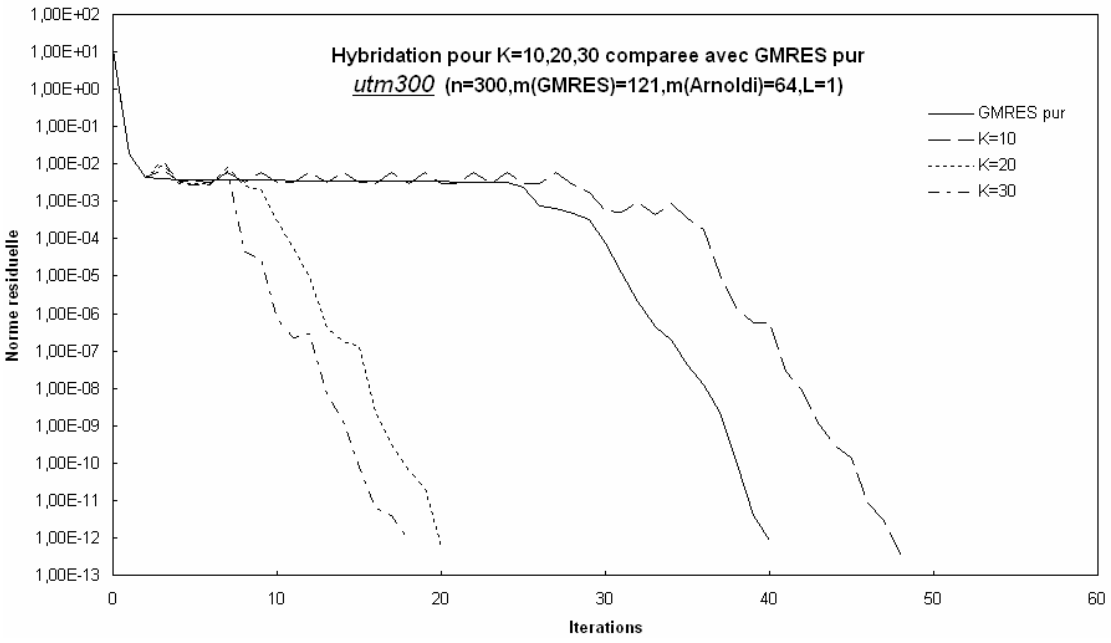
Type 2, α est un vecteur « 1,2, …, QUOTA » avec la taille « QUOTA ».

Type 3, α est un vecteur inverse de type 1 « QUOTA, QUOTA-1, …, 1 » avec la taille « QUOTA ».

Type 4, α est un vecteur composé par les vecteurs résiduels de GMRES avec la taille « QUOTA ».

Type 5, α est un vecteur composé par les vecteurs résiduels de GMRES, mais les $\frac{Min(res)}{res}$ éléments sont avec la taille « QUOTA ».

On peut remarquer que pour la matrice « utm300 » le type 5 est plus efficace, mais pour la matrice « utm1700a », c'est le type 3 (voir FIG 4.18,FIG 4.19). Néanmoins, l'influence de ces types de redémarrage est moins grande que celle d'autres paramètres, cela dépend aussi beaucoup des matrices. D'autre part, les stratégies utilisées ici sont classiques. Sur FIG 4.18, les temps de calcul correspondant au type 5 sont plus grands que son nombre d'itérations le suggère, comparé aux autres. Ceci est dû à la charge de la machine lors des calculs correspondant, chaque itération fut plus longue pour cet essai là.



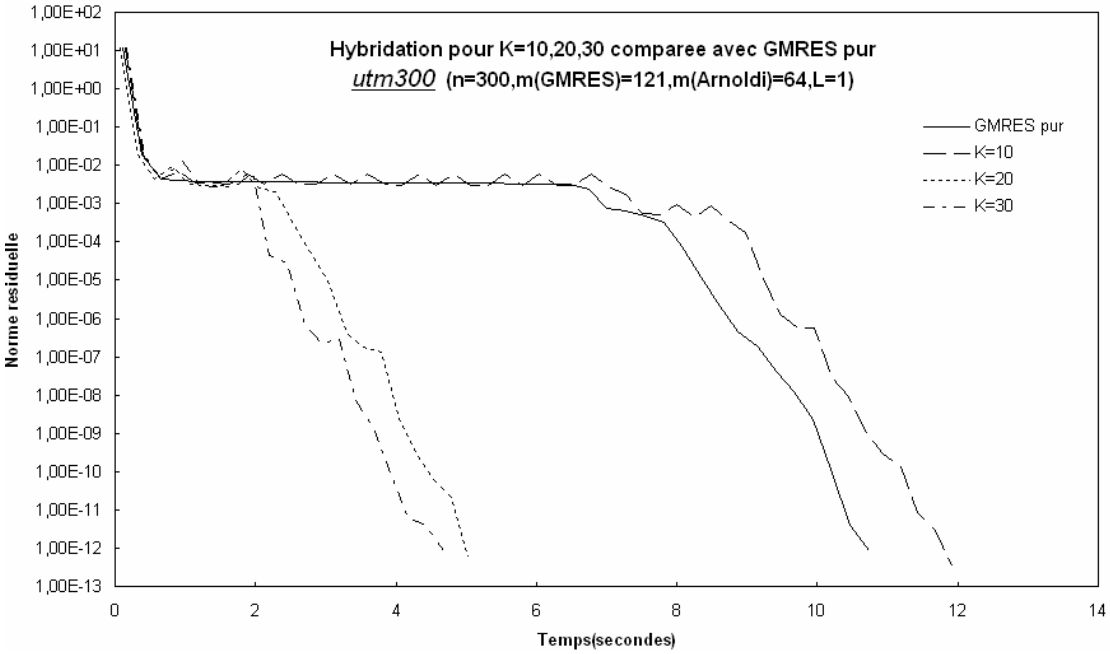


FIG 4.11 L'influence de K (matrice utm300)

Dans cette figure FIG 4.11, le temps permettant d'atteindre la convergence de la méthode hybride (***k***=20,30) est plus court que la méthode traditionnelle GMRES(***m***) pour la matrice « *utm300* ». Pourtant pour les valeurs de ***k*** petites, l'efficacité de la méthode hybride peut s'avérer insuffisante puisque le nombre d'itérations nécessaires à la convergence peut dépasser celui de la méthode traditionnelle GMRES(***m***). C'est le cas ici pour ***k***=10. Au contraire, un degré suffisant du polynôme « Least Squares » (ici ***k***=20 et ***k***=30) à la fois un nombre d'itérations et un temps de calcul avant convergence très avantageuse.

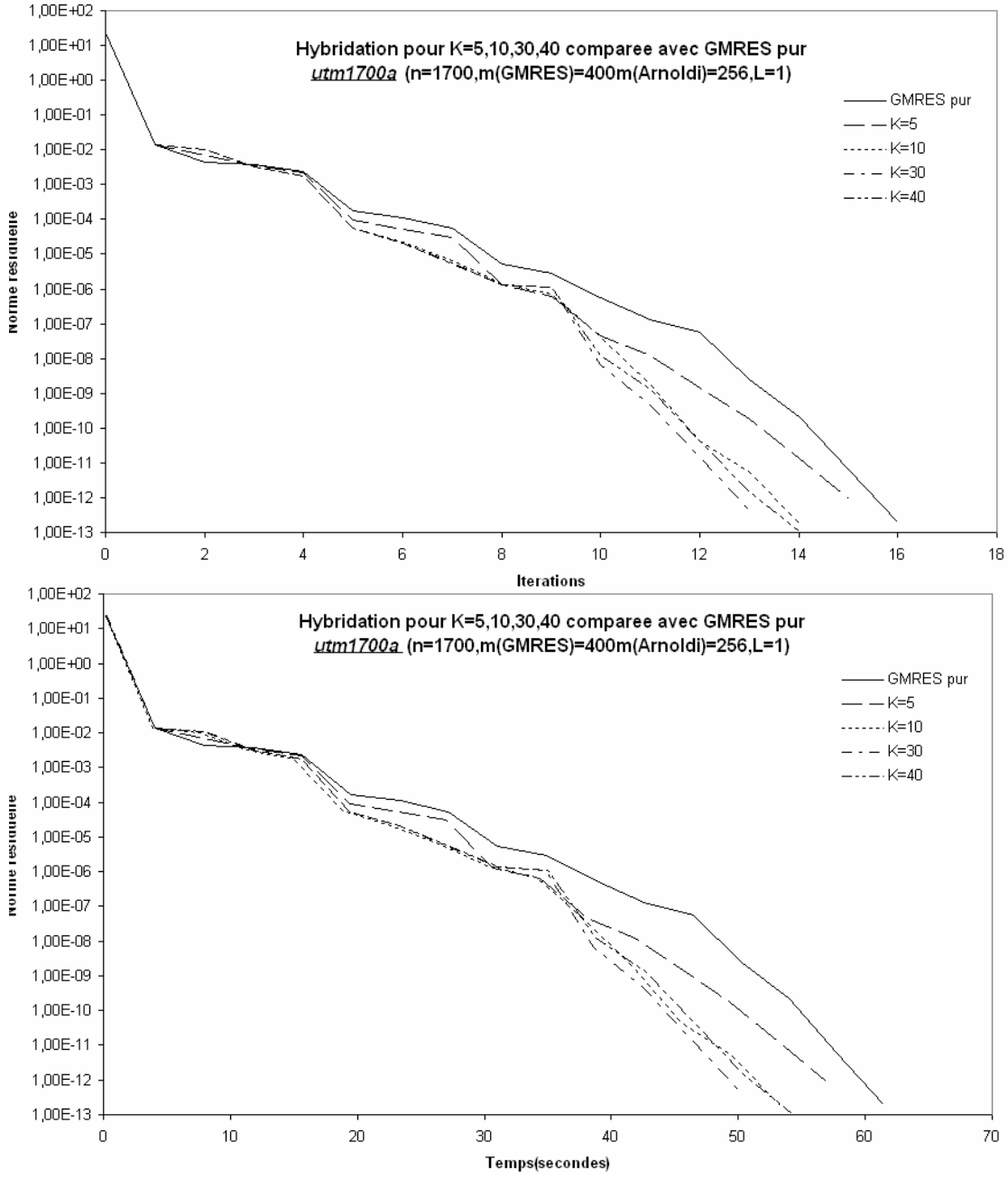


FIG 4.12 - L'influence de K (matrice utm1700a)

Dans cette figure FIG 4.12, le temps permettant d'atteindre la convergence par la méthode hybride est systématiquement plus court que par la méthode traditionnelle GMRES(***m***) pour la matrice « *utm1700a*» quelle que soit la valeur de ***k***. Quand ***k*** s'accroît de 5 à 30, le temps de convergence devient de plus en plus court. Mais quand on augmente la valeur ***k*** de 30 à 40, le temps de convergence s'accroît, au contraire. On trouve dans ce cas une valeur optimale de ***k*** qu'est 30.

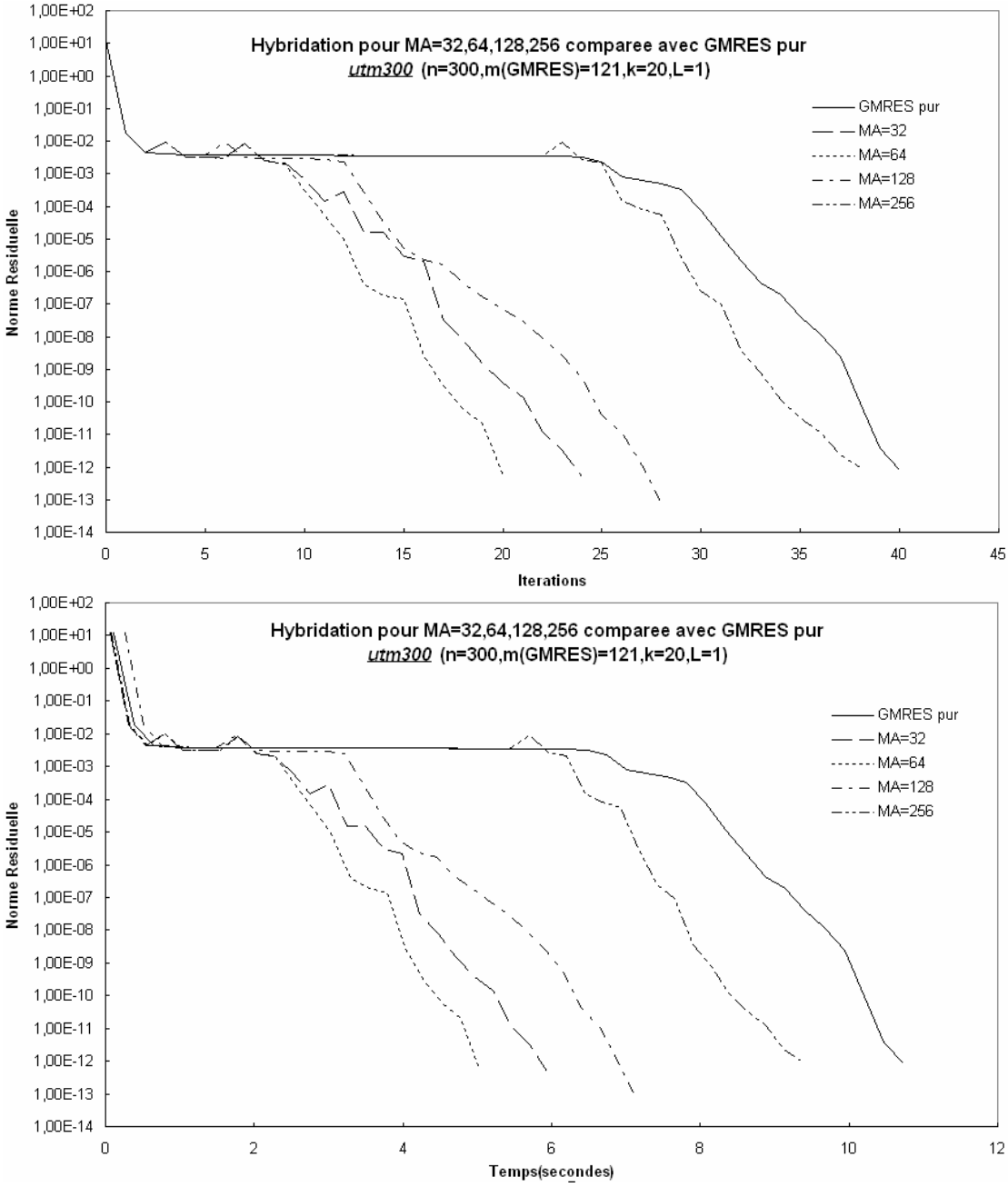


FIG 4.13 - L'influence de MA (matrice utm300)

Dans cette figure FIG 4.13, le temps permettant d'atteindre la convergence de la méthode hybride est plus court que par la méthode traditionnelle GMRES(***m***) pour la matrice « *utm300*». Quand ***MA*** s'accroît de 32 à 64, le temps de convergence devient de plus en plus court. Mais quand on augmente la valeur ***MA*** de 64 à 256, le temps de convergence contrairement s'accroît. On trouve que dans ce cas la valeur optimale de ***MA*** est 64. En effet, lorsque ***MA*** augmente, le calcul des valeurs propres devient plus efficace et leur prise en compte intervient plus rapidement. Toutefois, le coût de leur calcul croît également avec ***MA***, d'où l'existence d'un optimum.

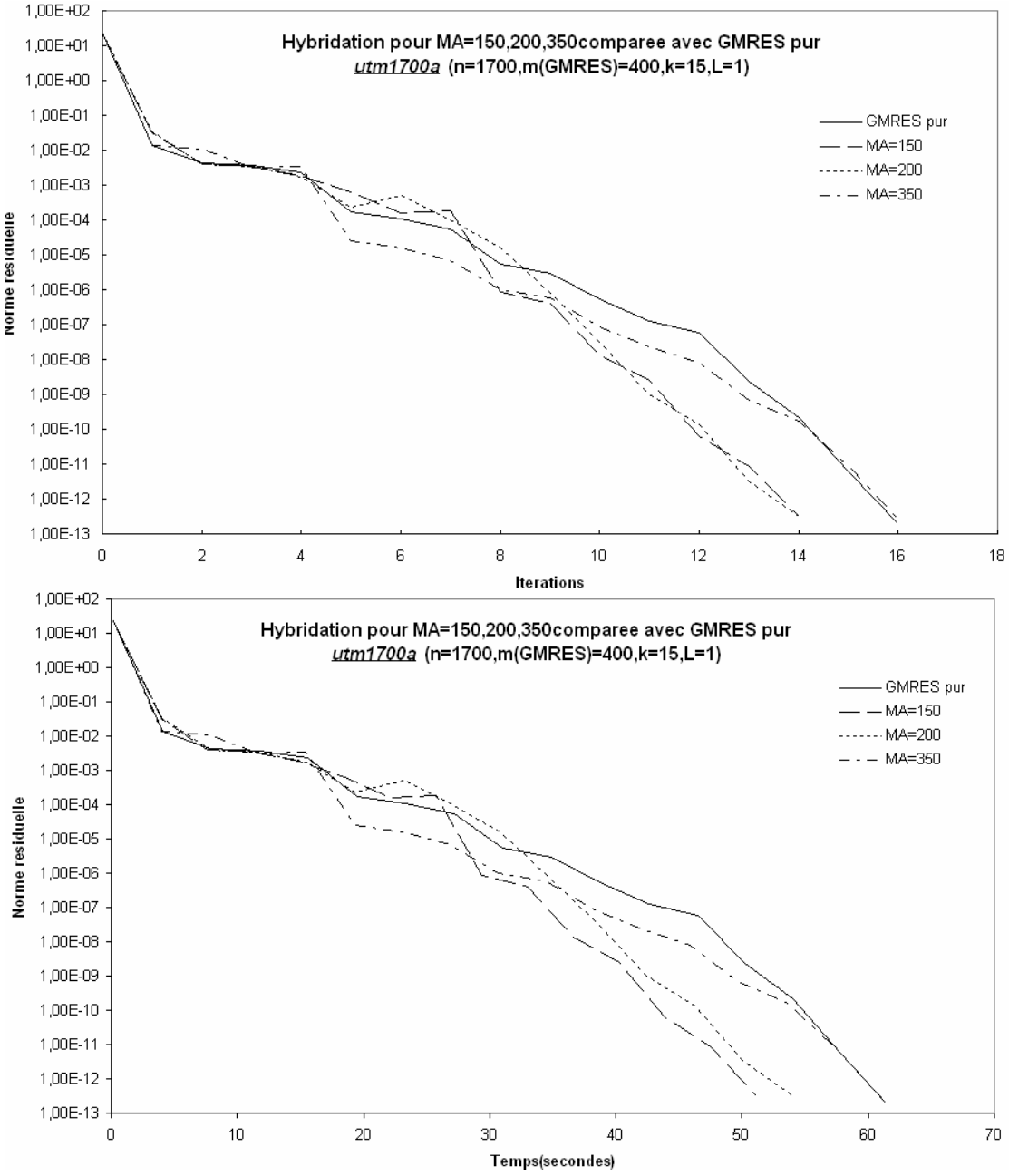


FIG 4.14-L'influence de MA (matrice utm1700)

Dans cette figure FIG 4.14, le temps permettant d'atteindre la convergence de la méthode hybride est plus court que par la méthode traditionnelle GMRES(***m***) pour la matrice « *utm1700a*». Quand ***MA*** s'accroît de 150 à 350, le temps de convergence devient de plus en plus long. On trouve que dans ce cas la valeur optimale de ***MA*** est la valeur la plus petite 150.

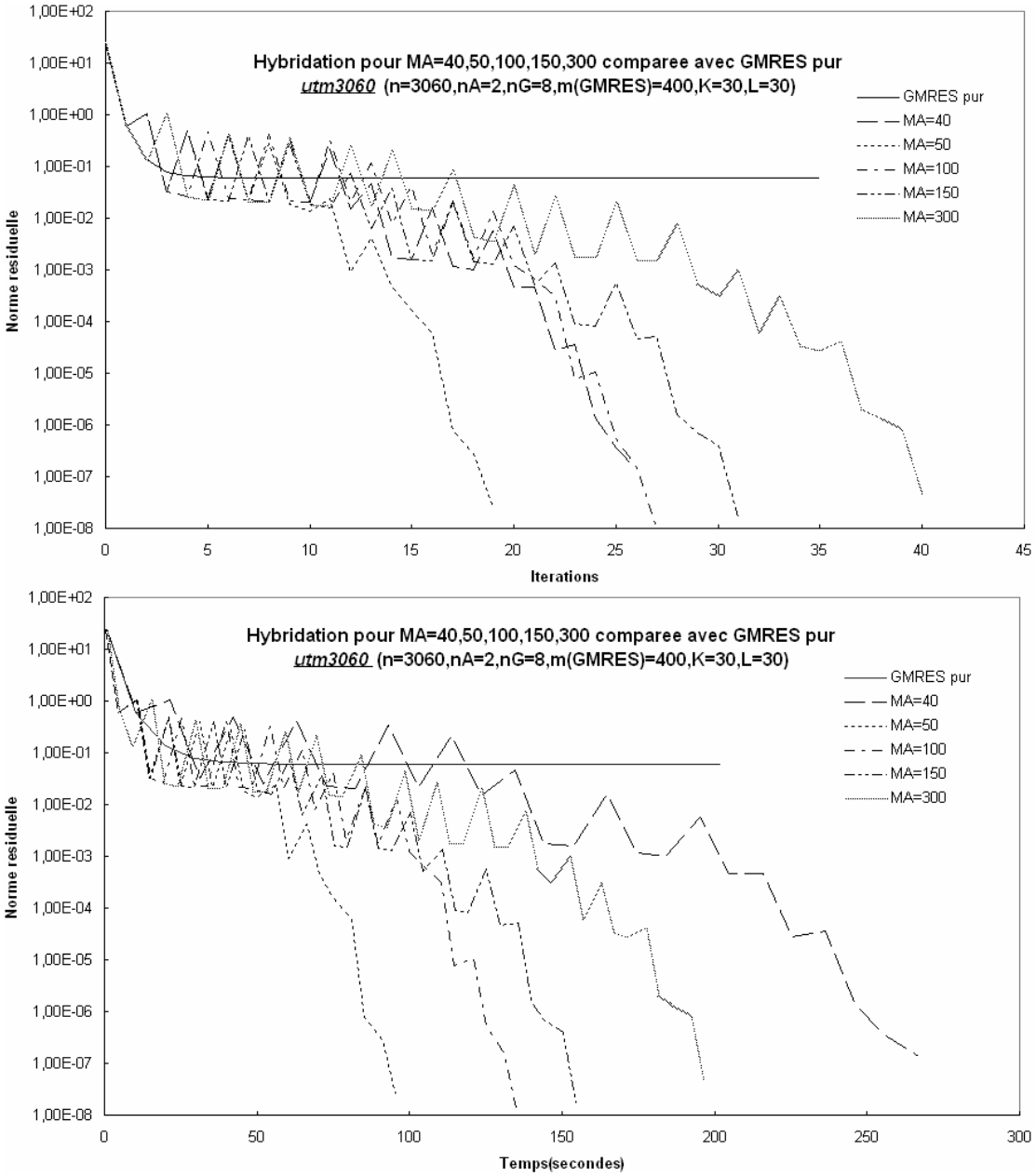


FIG 4.15 –L'influence de MA (matrice utm3060)

Dans cette figure FIG 4.15, le temps permettant d'atteindre la convergence de la méthode hybride est plus court que par la méthode traditionnelle GMRES(***m***) pour la matrice « *utm3060*». Quand ***MA*** s'accroît de 40 à 50, le temps de convergence devient de plus en plus court. Mais quand on augmente la valeur ***MA*** de 50 à 300, le temps de convergence contrairement s'accroît. On trouve que dans ce cas la valeur optimale de ***MA*** est 50.

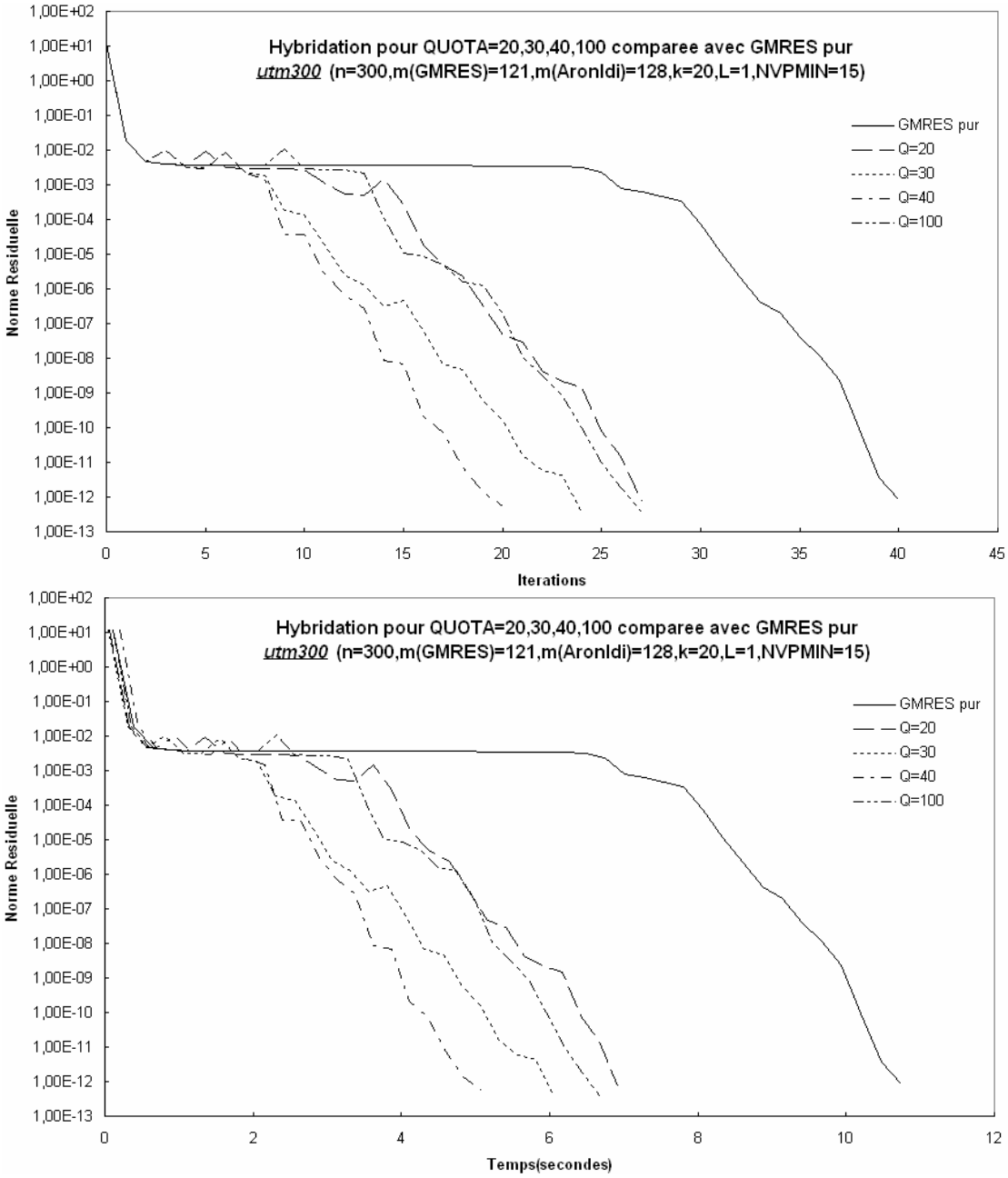


FIG 4.16 L'influence de QUOTA (matrice utm300)

Dans cette figure FIG 4.16, le temps permettant d'atteindre la convergence de la méthode hybride est plus court que la méthode traditionnelle GMRES(***m***) pour la matrice « *utm300*». Quand ***QUOTA*** s'accroît de 20 à 40, le temps de convergence devient de plus en plus court. Mais quand on augmente la valeur ***QUOTA*** de 40 à 100, le temps de convergence contrairement s'accroît. On trouve que dans ce cas la valeur optimale de ***QUOTA*** est 40. En effet, QUOTA est le paramètre qui détermine le nombre de valeurs propres à atteindre avant d'autoriser les prises en compte dans un calcul « Least Squares ». L'efficacité de cette prise en compte croît avec la valeur de QUOTA,

car alors les paramètres « Least Squares » sont mieux calculés. Mais cette valeur de QUOTA élevée espace les prises en compte, et rend donc l'hybridation moins performante.

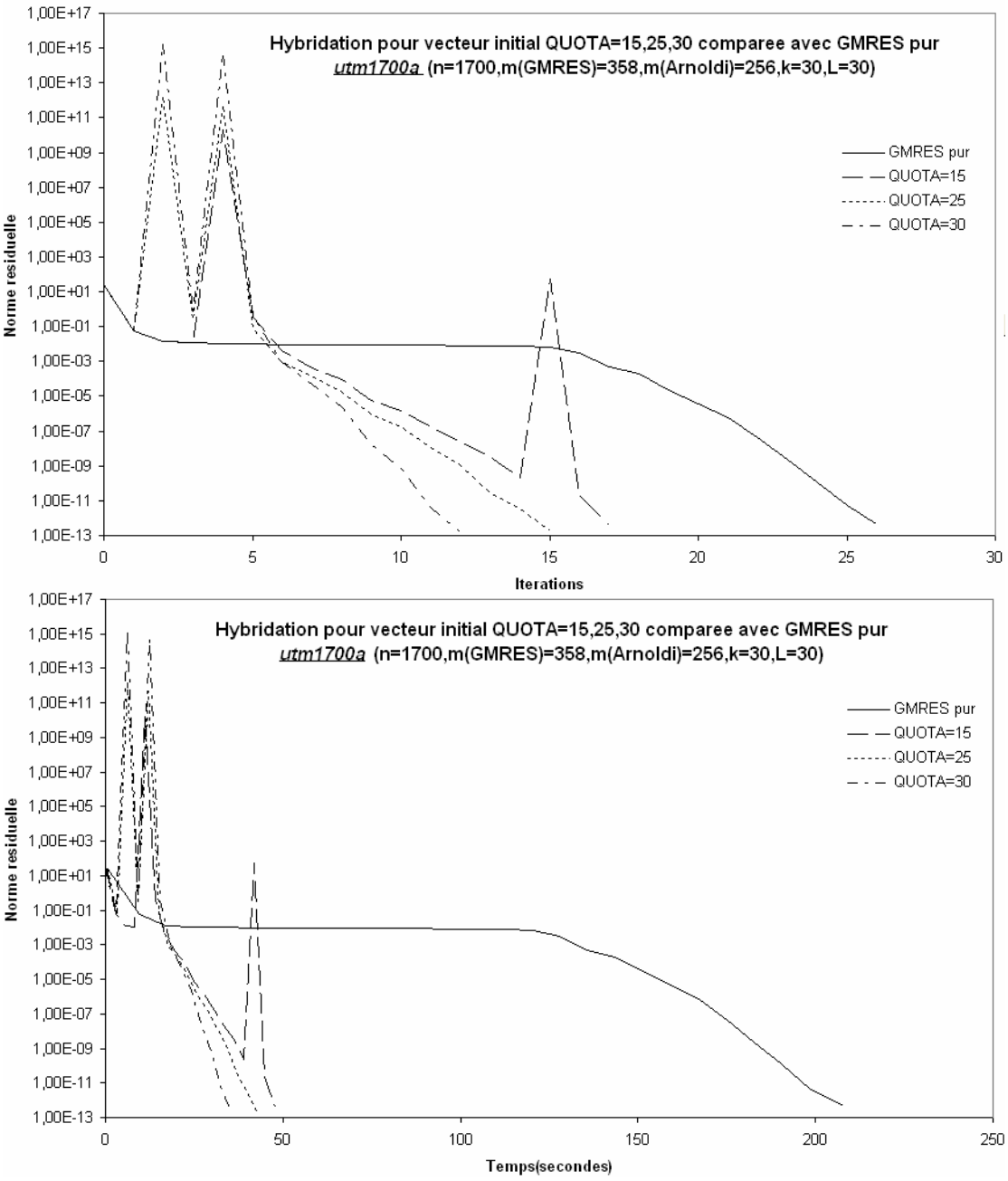


FIG 4.17 L'influence de QUOTA (matrice utm1700a)

Dans cette figure FIG 4.17, le temps permettant d'atteindre la convergence de la méthode hybride est plus court que par la méthode traditionnelle GMRES(***m***) pour la matrice « *utm1700a*». Quand ***QUOTA*** s'accroît de 15 à 30, le temps de convergence

devient de plus en plus court. On trouve que dans ce cas la valeur optimale de ***QUOTA*** est 30.

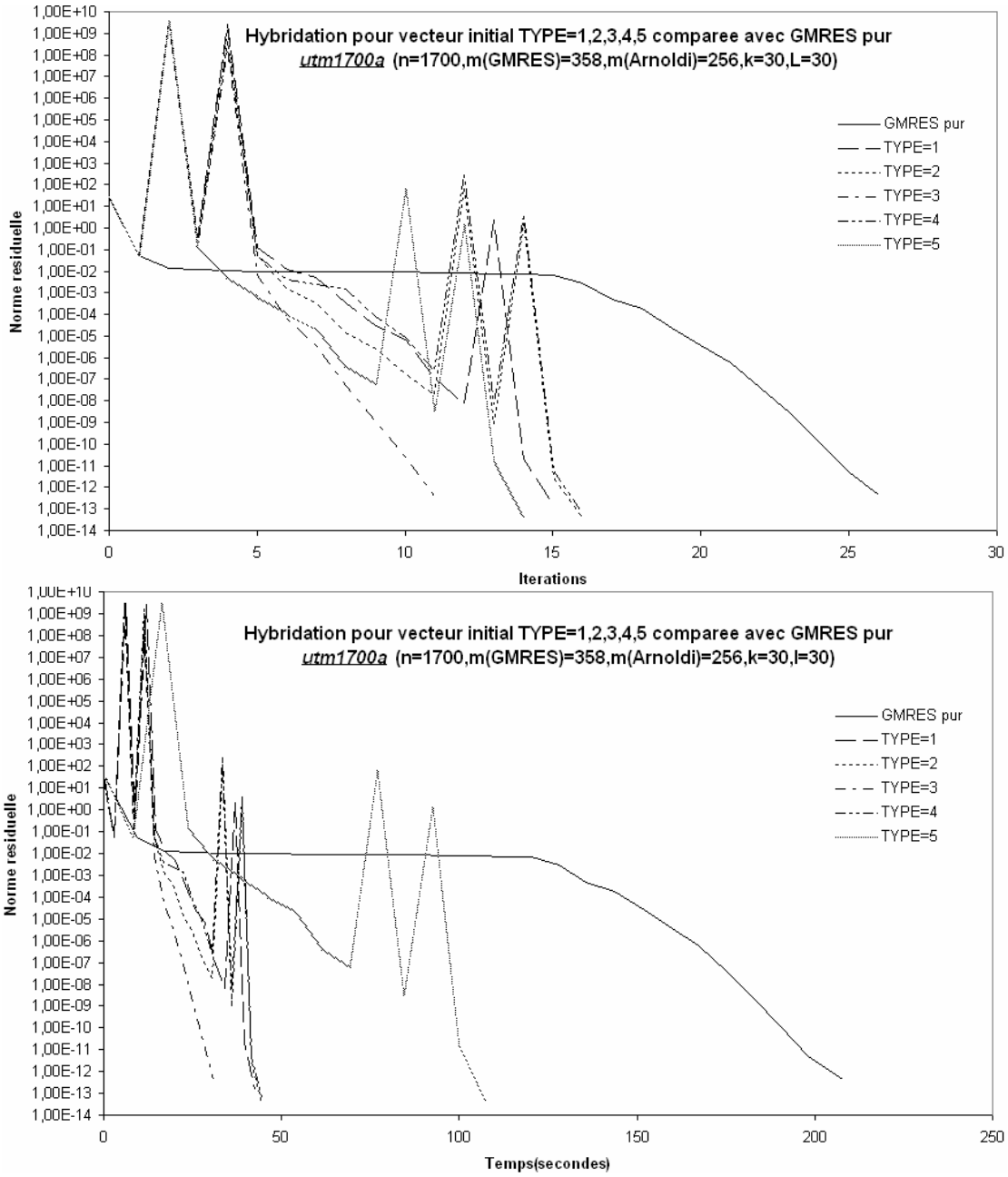


FIG 4.18 -L'influence de TYPE (matrice utm1700a)

Dans cette figure FIG 4.18, le temps permettant d'atteindre la convergence de la méthode hybride est plus court que par la méthode traditionnelle GMRES(***m***) pour la matrice « *utm1700a*». Pour cette matrice le vecteur initial TYPE3 est le plus efficace, Il est à noter que le choix de ce vecteur initial dépend étroitement de la matrice.

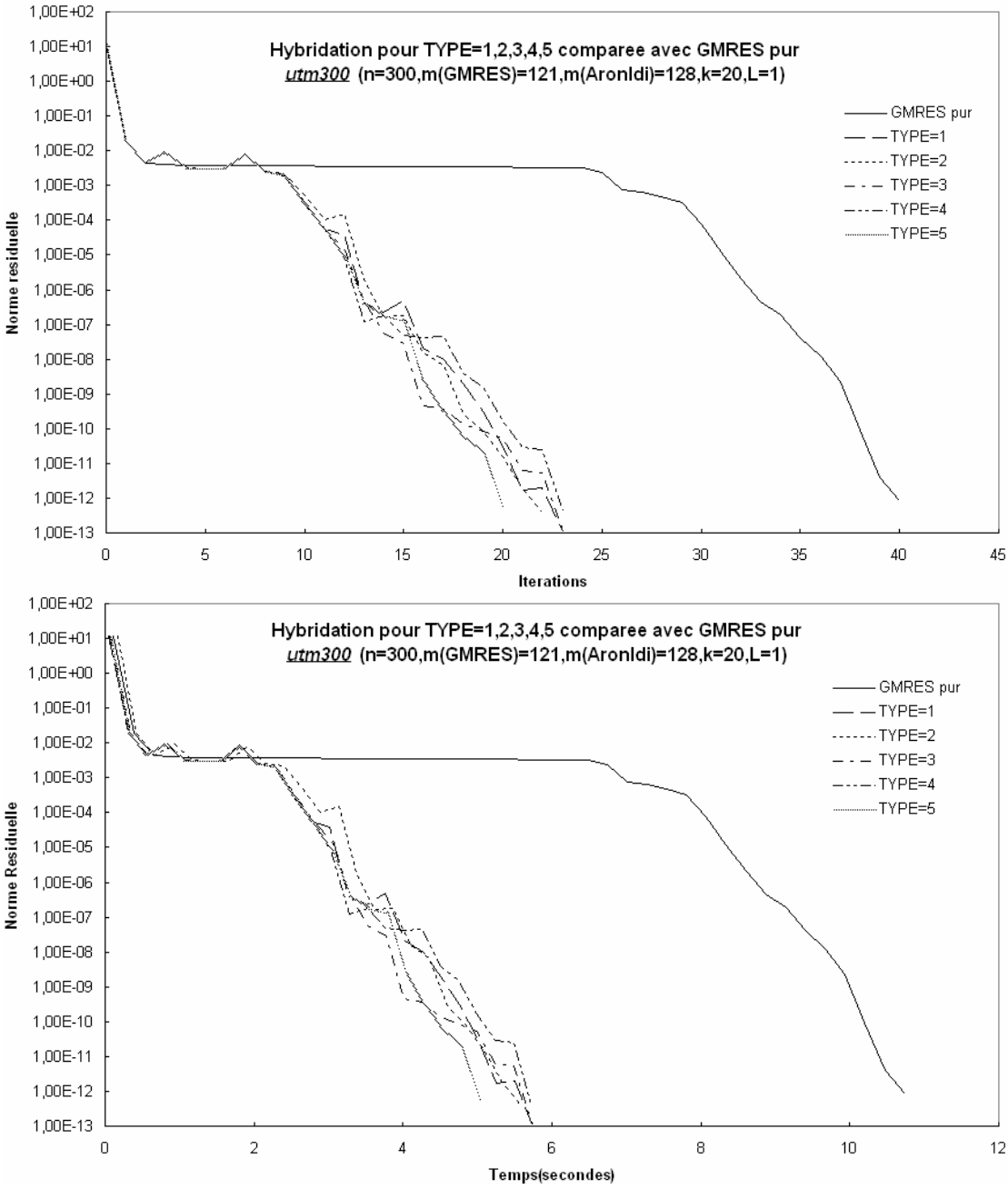


FIG 4.19 –L'influence de TYPE (matrice utm300)

Dans cette figure FIG 4.19, le temps permettant d'atteindre la convergence de la méthode hybride est plus court que par la méthode traditionnelle GMRES(***m***) pour la matrice « *utm300*». Pour cette matrice le vecteur initial TYPE5 est le plus efficace.

4.4.4 Influence de la précision des valeurs propres approchées

La précision des valeurs propres approchées par la méthode d'Arnoldi est un critère jouant sur plusieurs aspects de la convergence et des performances de la méthode hybride. Si cette précision est faible, nous pouvons rapidement envoyer des données au processus

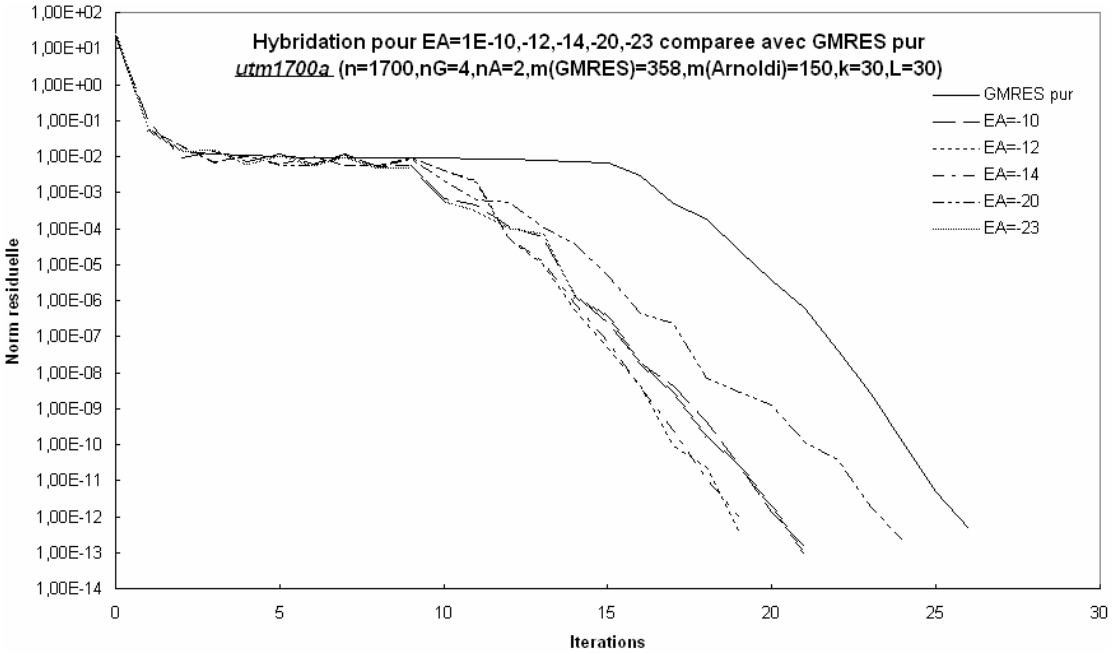
construisant le convexe et l'ellipse permettant de lancer des itérations de la méthode des moindres carrés. La méthode de GMRES sera donc « accélérée » plus tôt que si la précision sur les valeurs propres est plus exigeante. Néanmoins, cette accélération peut être moins bonne, a priori, que si on attend un petit peu plus mais que nous obtenons des valeurs plus proche de la valeur mathématique dans l'espace des complexes. On remarque que dans le cas de la matrice « utm3060 », la précision est plus faible, l'arrivée de la convergence est plus rapide (voir FIG 4.21).

Il est également possible de demander une précision faible dans un premier temps et de l'affiner au fur et à mesure. Par exemple, nous pouvons envoyer des valeurs propres approchées à ***Epsilon*** près pour commencer le procédé d'accélération puis de reprendre les calculs des valeurs propres en demandant une meilleure approximation, ***Epsilon/k*** par exemple (***k*** étant un entier supérieur à un). Lorsque les nouvelles valeurs propres sont obtenues, elles sont envoyées pour calculer une nouvelle ellipse qui sera utilisée lors des lancements des prochaines itérations de la méthode des moindres carrés. Tout ceci est calculé de manière asynchrone sans jamais être bloquant pour les procédés en cours de calcul. Nous avons effectué une série de tests pour analyser cela. Les figures *FIG 4.21*, FIG 4.22 montrent la convergence pour diverses valeurs de la précision demandée sur les valeurs propres approchées. Dans un des cas, nous demandons une précision absolue de 0.001 alors que dans l'autre cas nous exigeons une précision de 10^{-7} et 10^{-20}. Le troisième cas commence par demander une précision de 0.001 puis reprend les calculs de valeurs propres avec une précision dix fois plus petite. Ce procédé est reproduit jusqu'à avoir une précision de 10^{-7} et 10^{-20}. Théoriquement, il est possible de continuer ainsi jusqu'à la convergence de GMRES(***m***) puisque plus les valeurs propres sont précises, plus l'accélération est performante. De plus, a priori, il n'y a que très peu de retard dû à ces calculs en parallèle puisque cela est asynchrone et non bloquant. Néanmoins, la prise en compte des premières valeurs propres, détermine le nombre de fois que nous allons faire une méthode des moindres carrés, pour la résolution d'un système donné, à une précision donnée. En effet, plus tôt sont envoyées les valeurs propres aux autres processus, plus tôt nous commençons à lancer des méthodes des moindres carrées, en temps absolu. C'est pourquoi nous pouvons, comme dans la *FIG 4.21*, avoir des temps de calcul plus longs que d'autres tests qui font moins d'itérations de GMRES(***m***). Ceci s'explique par le fait que la prise en compte rapide de valeurs propres commence le procédé d'accélération plus rapidement que d'autres tests, ce qui permet donc de pouvoir parfois converger plus vite,

mais oblige à faire plus d'itération de la méthode des moindres carrées, qui ne sont pas comptabilisées ici. Par exemple, dans la *FIG 4.21* nous voyons que si une petite précision est meilleure pour la convergence, une précision plus grande est meilleure en terme de temps de calcul. Une précision de 0.001 permet de commencer l'accélération de GMRES(***m***) rapidement et donc d'obtenir une bonne convergence mais entraîne de nombreux lancement de processus des moindres carrés coûteux. Dans la FIG 4.23, on remarque que pour la matrice *utm1700a*, dans le cas EA=10^{-03}, cette matrice ne converge pas. Mais quand on diminue la précision de 10^{-03} à 10^{-20} par pas de 10, on atteinte la convergence et il est plus rapidement que dans le cas EA=10^{-20}.

D'autre part, l'indéterminisme et les nombreux paramètres à prendre en compte rendent difficile une analyse détaillée de la performance, Dans la FIG 4.22, FIG 4.23, *FIG 4.24* nous remarquons qu'une stratégie avec précision grandissant progressivement est un bon compromis pour minimiser le temps de calcul.

En fait, cela dépend des matrices, il n'est pas évident de trouver pour chacune d'elle la précision donnant la meilleure accélération. On marque que dans les FIG 4.20, FIG 4.21 , pour la matrice *utm1700a* EA=10^{-14} donne la meilleur résultat, mais pour la matrice utm3060, il s'agit de EA=10^{-3}.



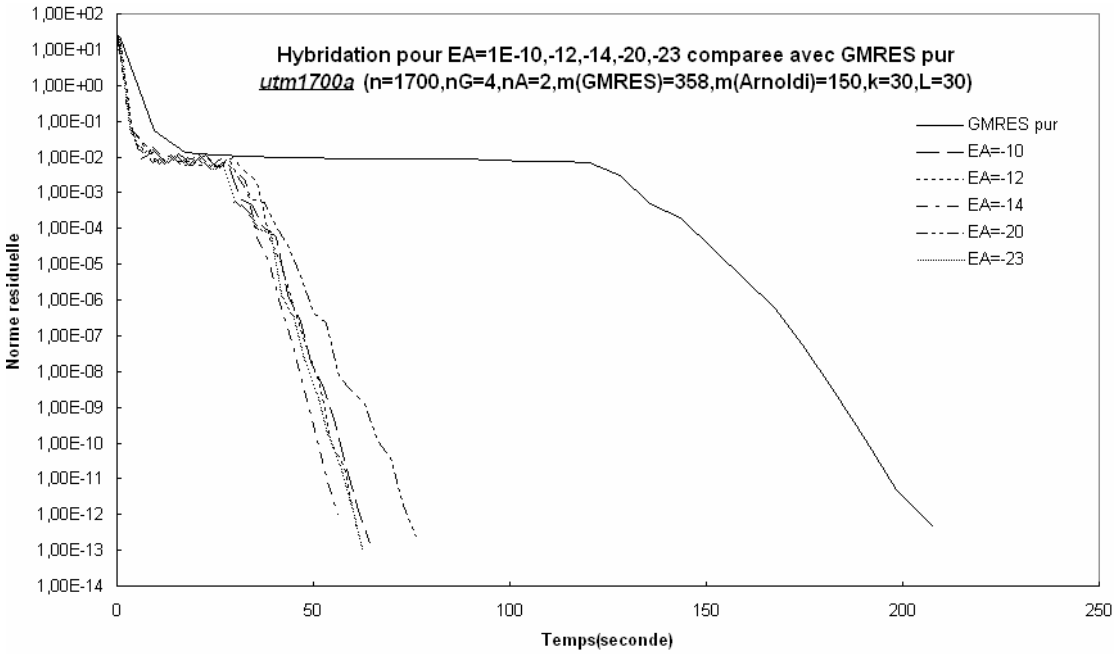


FIG 4.20-L'influence de EA (matrice utm1700a)

Dans cette figure FIG 4.20, le temps permettant d'atteindre la convergence de la méthode hybride est plus court que par la méthode traditionnelle GMRES(***m***) pour la matrice « *utm1700a*». Quand ***EA*** décroît de 10^{-10} à 10^{-20}, le temps de convergence devient de plus en plus court. Mais quand on diminue la valeur ***EA*** de 10^{-20} à 10^{-23}, le temps de convergence, au contraire, s'accroît. On trouve que dans ce cas la valeur optimale de ***EA*** est 10^{-20}, en effet, des valeurs propres plus précises accroissent l'efficacité de l'hybridation, car les paramètres « Least Squares » sont alors mieux calculés. Mais augmenter cette précision accroît également la durée de calcul des valeurs propres, ce qui espace les prises en compte et donc diminue les performances de la méthode, d'où un optimum.

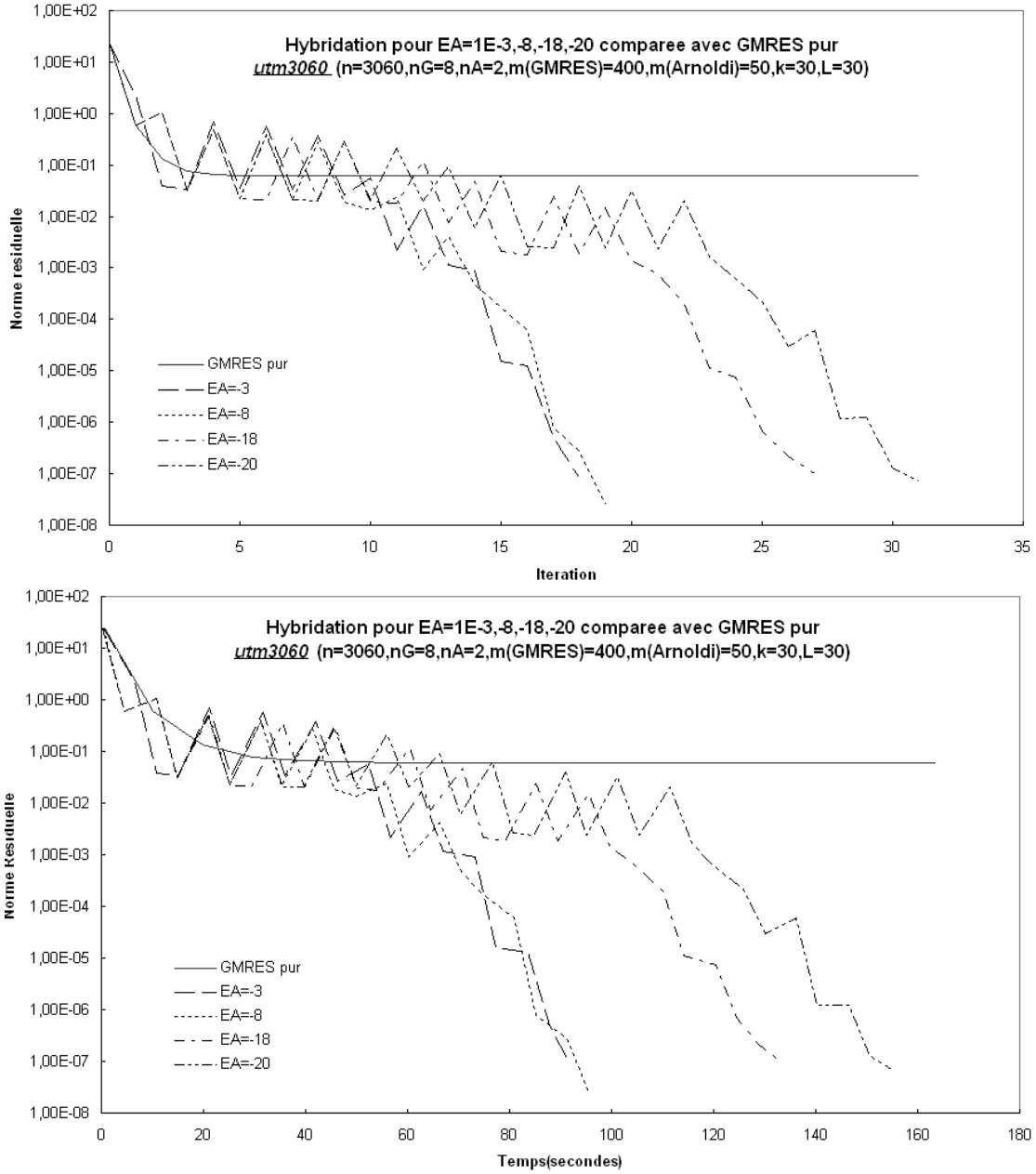


FIG 4.21 –L'influence de EA (matrice utm3060)

Dans cette figure FIG 4.21, le temps d'atteindre la convergence de la méthode hybride est plus court que la méthode traditionnelle GMRES(***m***) pour la matrice « *utm3060*». Quand ***EA*** se décroît de 10^{-3} à 10^{-20}, le temps de convergence devient de plus en plus long. On trouve que dans ce cas la valeur optimale de ***EA*** est 10^{-3}.

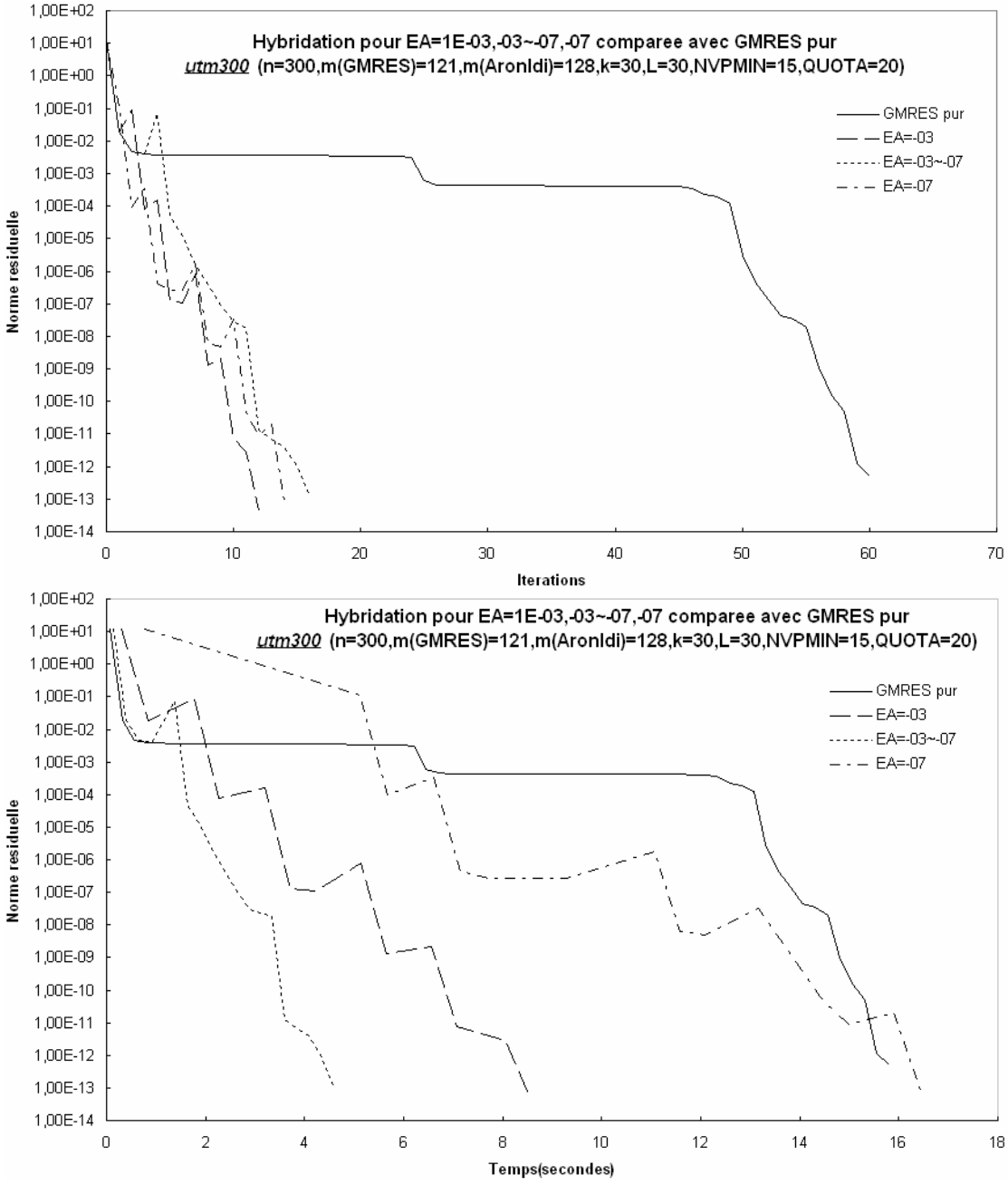


FIG 4.22 L'influence de EA (matrice utm300)

Dans cette figure FIG 4.22, le temps permettant d'atteindre la convergence de la méthode hybride est plus court que par la méthode traditionnelle GMRES(***m***) pour la matrice « *utm300*». Quand ***EA*** se décroît de 10^{-03} à 10^{-07}, le temps de convergence devient de plus en plus court.Mais surtout quand on diminue la valeur ***EA*** de 10^{-03} à 10^{-07} par pas de 10, la convergence est la plus rapide. On trouve que dans ce cas la valeur optimale de ***EA*** qui diminue de 10^{-3} à 10^{-7} par pas de 10.

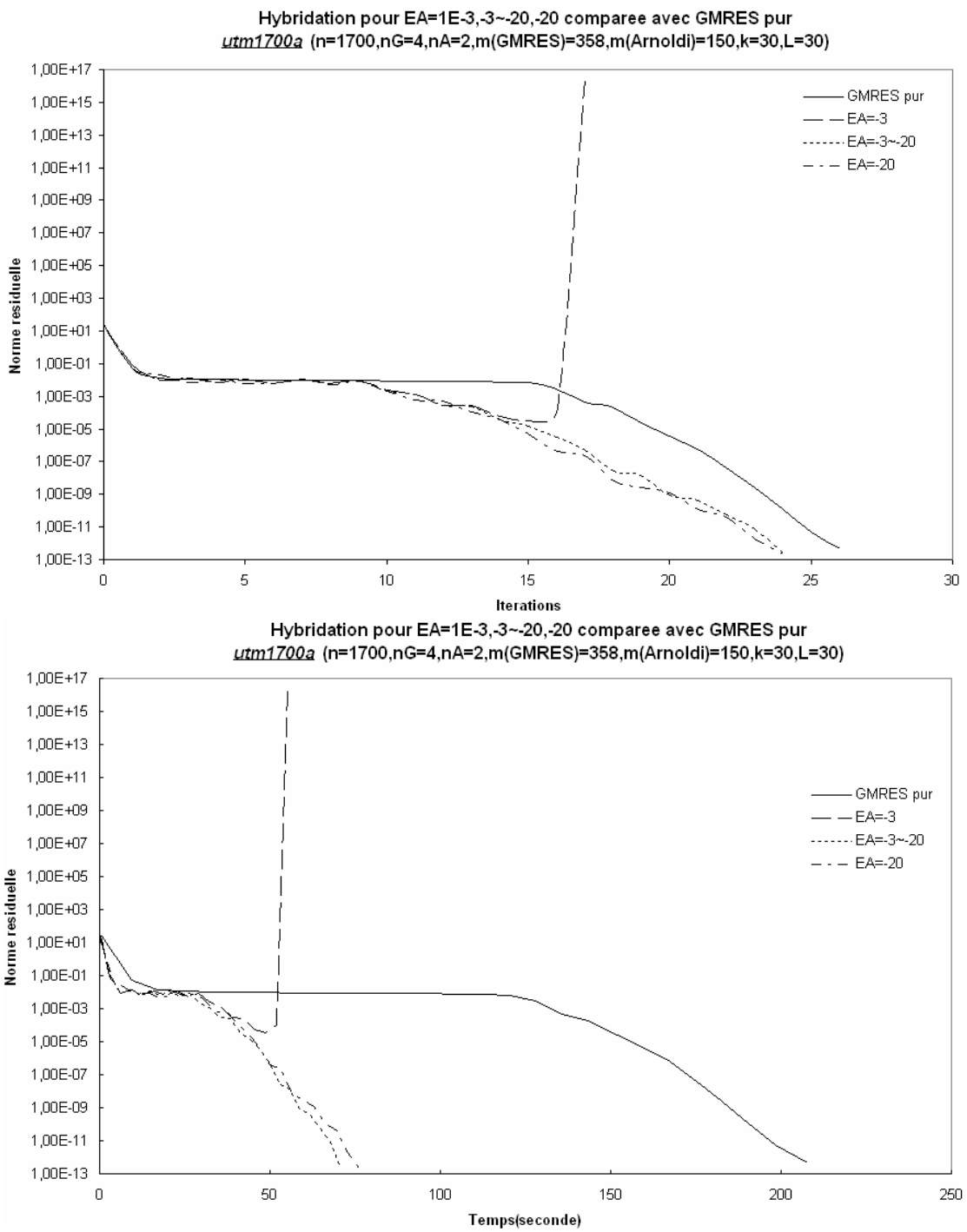


FIG 4.23 L'influence de EA (matrice utm1700a)

Dans cette figure FIG 4.23, le temps permettant d'atteindre la convergence de la méthode hybride (EA=$10^{-03\sim-20}$, 10^{-20}) est plus court que la méthode traditionnelle GMRES(***m***) pour la matrice « *utm1700a*». Quand la valeur ***EA*** est 10^{-03}, cette matrice ne converge pas d'où l'importance d'une précision suffisante du calcul des valeurs propres. Mais surtout quand on diminue la valeur ***EA*** de 10^{-03} à 10^{-20} par pas de 10, la

convergence est plus rapide. On trouve que dans ce cas la stratégie optimale pour ***EA*** est de diminue progressivement sa valeur de 10^{-3} à 10^{-20} par pas de 10.

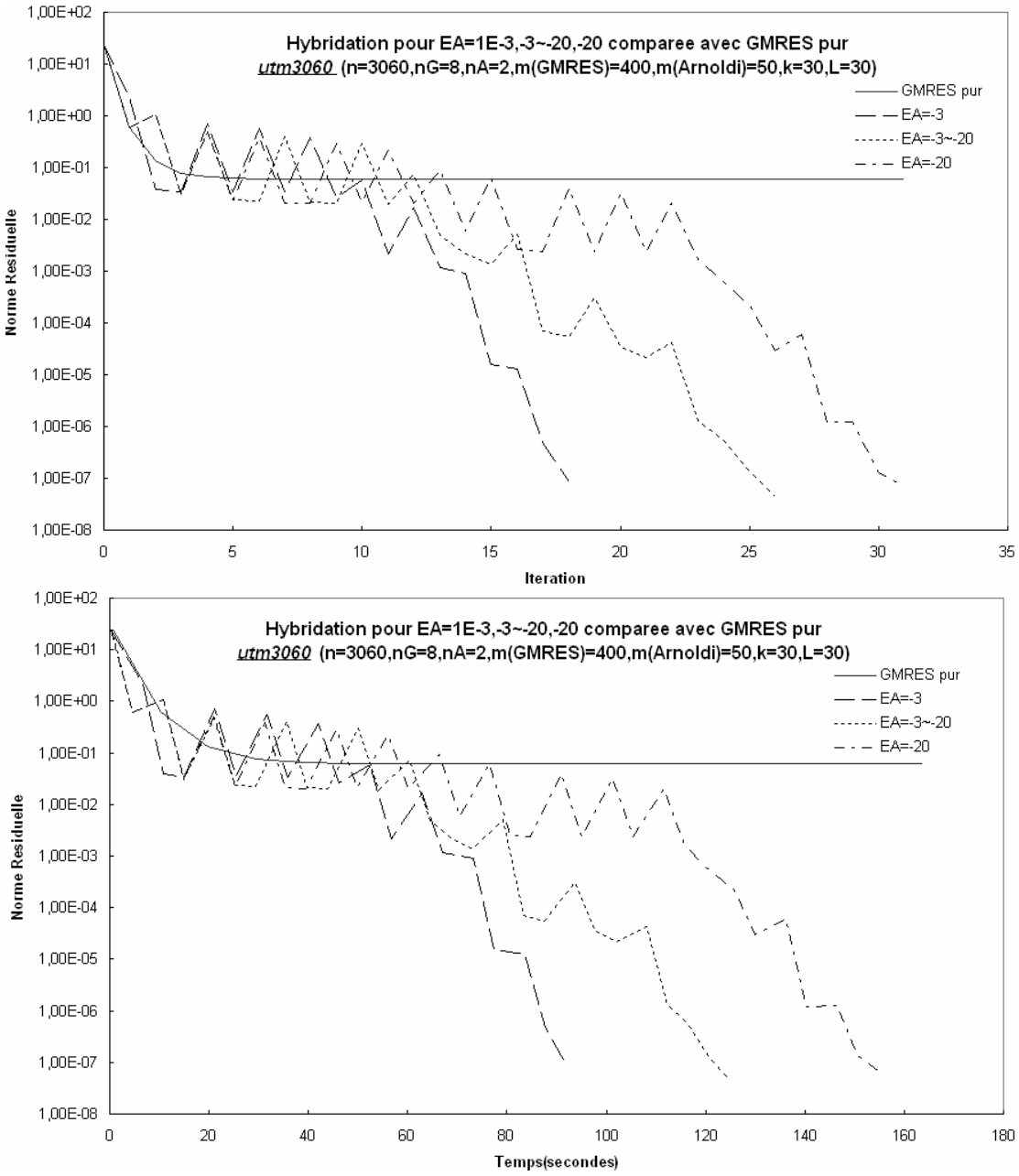


FIG 4.24 - L'influence de EA (matrice utm3060)

Dans cette figure FIG 4.24, le temps permettant d'atteindre la convergence de la méthode hybride est plus court que par la méthode traditionnelle GMRES(***m***) pour la matrice « *utm3060*». Même quand la valeur ***EA*** est 10^{-03}, cette matrice converge plus rapidement.Mais nous remarquons que quand on diminue la valeur de ***EA*** de 10^{-03} à 10^{-20}

par pas de 10, la convergence est plus rapide que le cas « $\boldsymbol{EA}=10^{-20}$ » et moins rapide que le cas « $\boldsymbol{EA}=10^{-03}$ ». On trouve que dans ce cas la valeur ***EA*** optimale est 10^{-03}.

4.4.5 Influence du nombre de processeurs pour GMRES ou ARNOLDI

Nous avons fait varier le nombre de processeurs attribués au calcul de GMRES (voir FIG 4.25 et FIG 4.26*)*, Nous remarquons que c'est avec le plus petit nombre de processeurs que l'on ne converge pas. Cela s'explique par le fait que quand le nombre de processeurs exécutant GMRES(*m*) est trop réduit comme ici, où seuls deux processeurs sont attribués au calcul de GMRES, mais avec une matrice moyenne (1700×1700), les communications parmi les processeurs exécutant GMRES(*m*) et celles entre le groupe exécutant PARPACK et le groupe exécutant GMRES(*m*) fonctionne très lentement à cause de ce moindre nombre. Quant à la matrice plus grande (*utm3060*), quand même six processeurs attribués au calcul de GMRES ne suffisent pas. Dans les cas convergents, nous remarquons que c'est avec le plus petit nombre de processeurs que l'on converge le plus vite. Cela s'explique par le fait que avec les grandes matrices le temps de communication parmi les processeurs a l'influence plus importante. Quand on augmente le nombre de processeurs consacrés au calcul de GMRES, l'efficacité diminue au contraire i.e. on consomme plus de temps pour atteindre la convergence parce que le temps de communication augmente beaucoup plus que celui de calcul que l'on a gagné. On remarque aussi que dans le cas de temps plus court, le nombre d'itérations n'est pas le plus petit. Cela s'explique par le fait que pour dans les cas convergents et le petit nombre de processeurs consacrés au calcul de GMRES, quand même le nombre d'itérations est plus important, le temps entre deux itérations est plus court, et le temps total est plus court.

Nous avons ensuite fait varier le nombre de processeurs attribués aux calculs des valeurs propres par le logiciel PARPACK (*FIG 4.27*). La convergence est d'autant meilleure que le nombre de processeurs est grand, nous pouvons donc conclure que plus nous avons de processeurs et plus nous produisons des valeurs propres aux processus de calcul des moindres carrés pour accélérer au plus tôt la méthode GMRES(***m***). Par contre, la variation des temps de calcul est très faible.

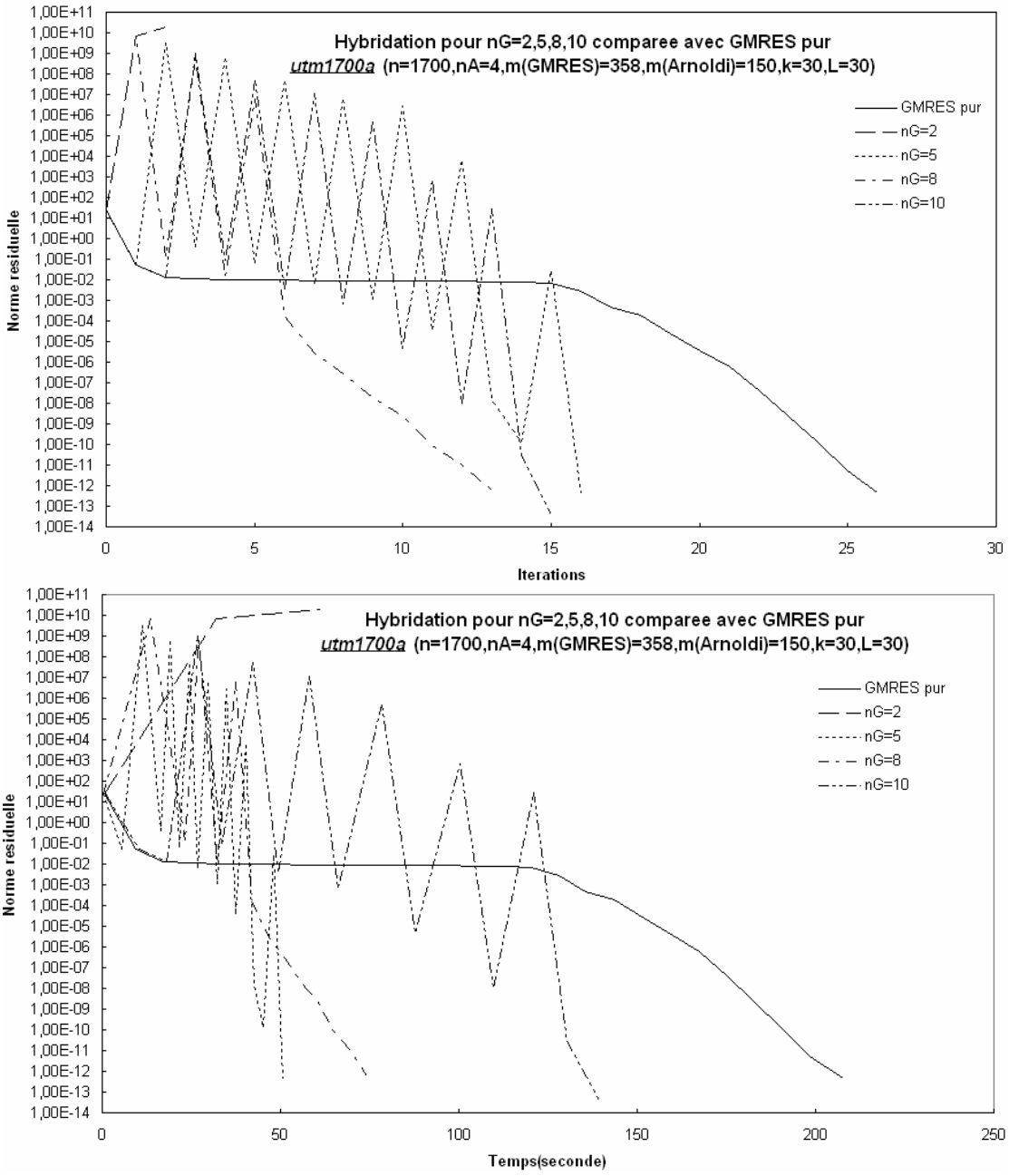


FIG 4.25 L'influence de nombre de processeurs pour GMRES (matrice utm1700a)

Dans cette figure FIG 4.25, le temps permettant d'atteindre la convergence de la méthode hybride (***nG***=5,8,10) est plus court que par la méthode traditionnelle GMRES(***m***) (***nG***=5) pour la matrice « *utm1700a*». Quand la valeur ***nG*** est 2, cette matrice ne converge pas. Mais nous remarquons quand on augmente la valeur ***nG*** de 5 à 10, la convergence est de plus en plus lente. Nous retrouvons ici les conséquences intervenant au sein de la méthode GMRES(***m***) parallèle. On trouve que dans ce cas la valeur ***nG*** optimale est 5.

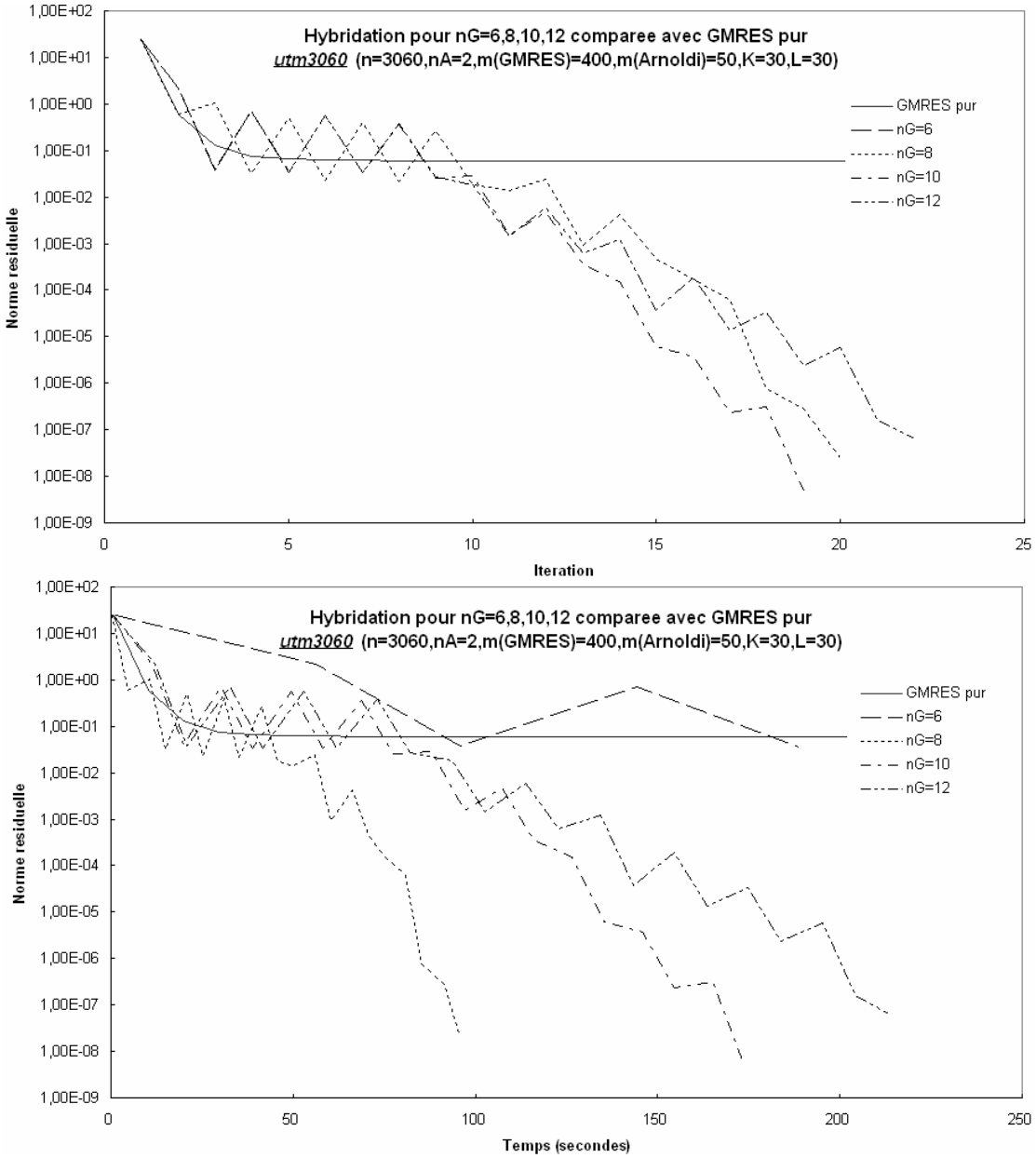


FIG 4.26 L'influence de nombre de processeurs pour GMRES (matrice utm3060)

Dans cette figure FIG 4.26, la convergence de la méthode hybride (***nG***=8,10,12) est très meilleure que la méthode traditionnelle GMRES(***m***) (***nG***=6) qui ne converge pas pour la matrice « *utm3060*». Quand la valeur ***nG*** est 6, cette matrice ne converge pas encore. Mais nous remarquons que quand on augmente la valeur ***nG*** de 8 à 12, la convergence est de plus en plus lente. On trouve que dans ce cas la valeur ***nG*** optimale est 8.

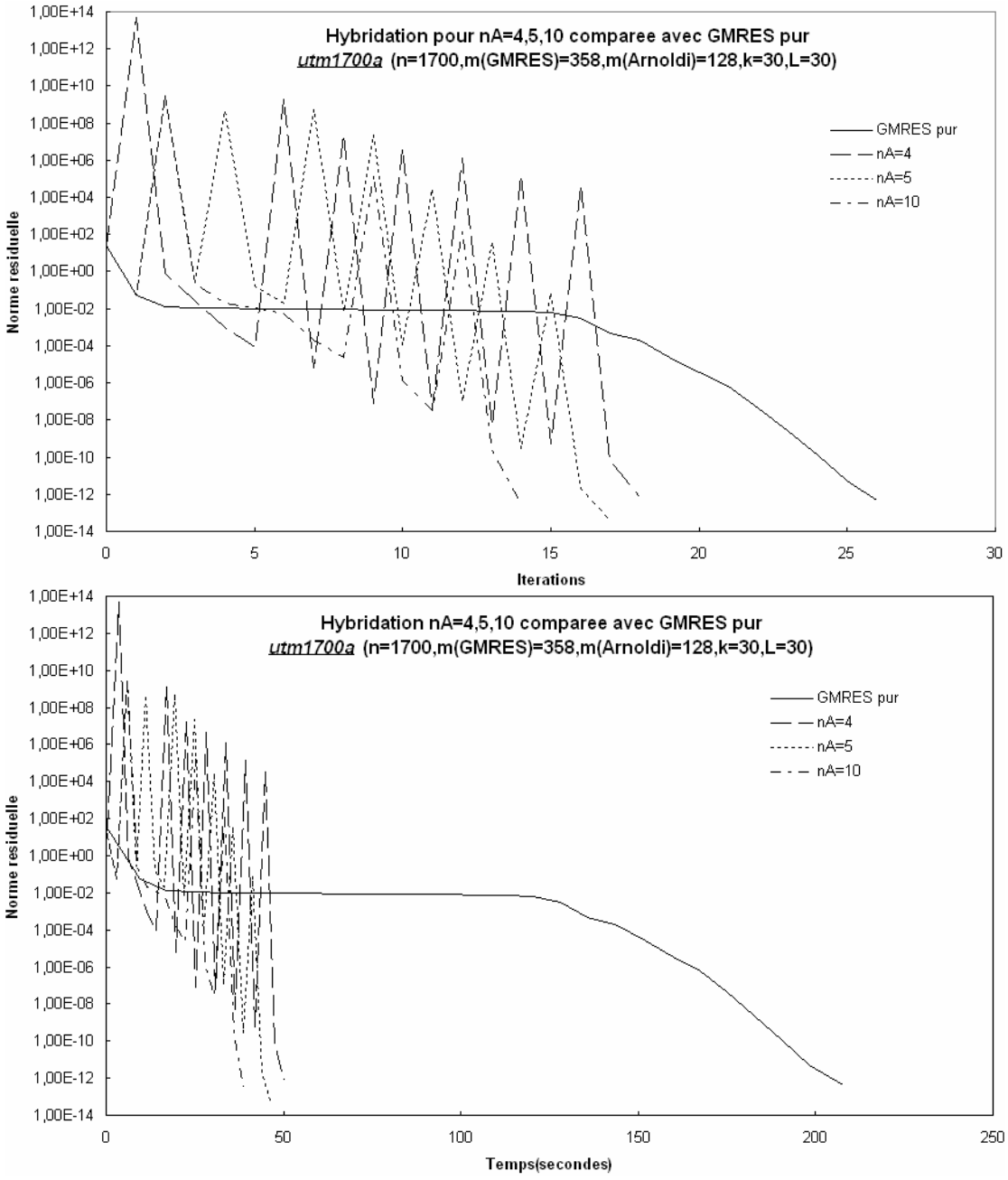


FIG 4.27- L'influence de nombre de processeurs pour Arnoldi (matrice utm1700a)

Dans cette figure FIG 4.27, le temps permettant d'atteindre la convergence de la méthode hybride est plus court que par la méthode traditionnelle GMRES(***m***) pour la matrice « *utm1700a*». Nous remarquons que quand on augmente la valeur ***nA*** de 4 à 10, la convergence est de plus en plus rapide. On trouve que dans ce cas la valeur ***nA*** optimale est 10. En effet, la méthode d'Arnoldi est, elle aussi extrêmement communicante et sa parallélisation présente un optimum.

4.5 Conclusion

Les résultats que nous avons obtenu lors des expérimentations révèlent que la méthode hybride asynchrone procure, comparée avec la méthode GMRES(***m***) classique, des accélérations de convergence remarquables, bien que des pics temporaires, parfois très élevés, soient montrés par l'évolution du résidu au moment de la prise en compte des valeurs propres. Ces accélérations et ces pics varient beaucoup d'une matrice sur l'autre, et reposent sur de nombreux paramètres dont il n'est pas évident de décider des valeurs les plus efficaces.

Du point de vue informatique, nous pouvons mieux tirer parti du parallélisme disponible en utilisant cette méthode, tout en conservant la granularité optimale des algorithmes impliqués.

Cette implantation réalisée sur une seule machine parallèle est particulièrement avantageuse, en permettant un déroulement indépendant et simultané de ces algorithmes dont les résultats numériques sont pris en compte sans délai d'attente, mais au prix d'une progression globale des valeurs calculées non déterministe (sauf en mode « batch ») mais convergente.

Néanmoins, le point délicat de cette méthode hybride reste le choix des divers paramètres. Nous remarquons une grande dépendance de la matrice quant au comportement de cette méthode. Nous avons montré de nombreux résultats permettant de s'apercevoir que cela n'est pas toujours simple mais que nous pouvons dégager des grandes lignes qui peuvent conduire à une expertise. Même si nous n'arrivons pas à trouver les paramètres optimaux pour une matrice donnée, nous pouvons espérer obtenir une bonne accélération et un temps de calcul moindre en utilisant cette méthode, même par une approche empirique de son optimisation.

Chapitre 5 Cas de matrices complexes

Plusieurs systèmes linéaires ont été testés pour la méthode GMRES (m) pour variables complexes. Les calculs ont toujours été réalisés sur la machine IBM RS6000/SP3 et SP4, que nous avons présentée en section 2.3.

5.1 Transformation de l'espace complexe à l'espace réel

On remplace la matrice carrée à variables complexes **A** par deux matrices carrées réelles de même dimension **AR** et **AI**, **AR** est la partie réelle des variables complexes de **A**, et **AI** est la partie imaginaire des variables complexes de **A**. Alors on peut donner la formule suivante :

$$\mathbf{A = AR + i * AI}$$

On transforme également les vecteurs complexes en deux vecteurs réels de même dimension. Par exemple :

$$\mathbf{x = xR + i * xI}$$

où **x** est un vecteur à variables complexes, **xR** (respectivement **xI**) est un vecteur à variables réelles qui représente la partie réelle (respectivement la partie imaginaire) des variables complexes de **x**.

Selon les transformations ci-dessus, on traduit les opérations de matrice et du vecteur de l'espace complexe dans l'espace réel. Par exemple, pour le produit matrice-vecteur, on aura

$$\mathbf{Ax = (AR + i * AI) * (xR + i * xI) = (AR * xR - AI * xI) + i * (AI * xR + AR * xI)}$$

Ainsi, nous pouvons obtenir le produit matrice-vecteur à variables complexes en calculant des produits matrice-vecteur à variables réelles.

5.2 Formats adaptés au cas complexe

Pour le cas complexe, on espère également gagner de la place en mémoire et du temps de calcul en gérant correctement les matrices creuses à variables complexes comme en section 3.1.

5.2.1 Format CSR

Le format **CSR** adapté au cas complexe décrit la matrice complexe creuse par quatre vecteurs (voir l'exemple de la FIG 5.1) :

AR est un vecteur contenant les parties réelles des éléments non nuls de la matrice, rangées ligne par ligne.

AI est un vecteur contenant les parties imaginaires des éléments non nuls de la matrice, rangées ligne par ligne. On met 0 dans le vecteur **AR** (respectivement **AI**) si la partie réelle (respectivement la partie imaginaire) des éléments non nuls est nulle. Ainsi **AI** est un vecteur de même taille que **AR**.

JA est un vecteur de même taille que **AR** (et **AI**) dont chaque élément contient le numéro de colonne de l'élément correspondant de **AR** (et **AI**).

IA est un vecteur de taille $n+1$, où n est le nombre de lignes de la matrice. Chaque élément **IA** (i) de ce vecteur ($i \in [1,n]$) contient le rang, dans le vecteur **AR** (et **AI**), du 1^{er} élément de la ligne n°i de la matrice. **IA (*n+1*)** contient T+1, où T est la taille de **AR** (ainsi que **AI** et de **JA**).

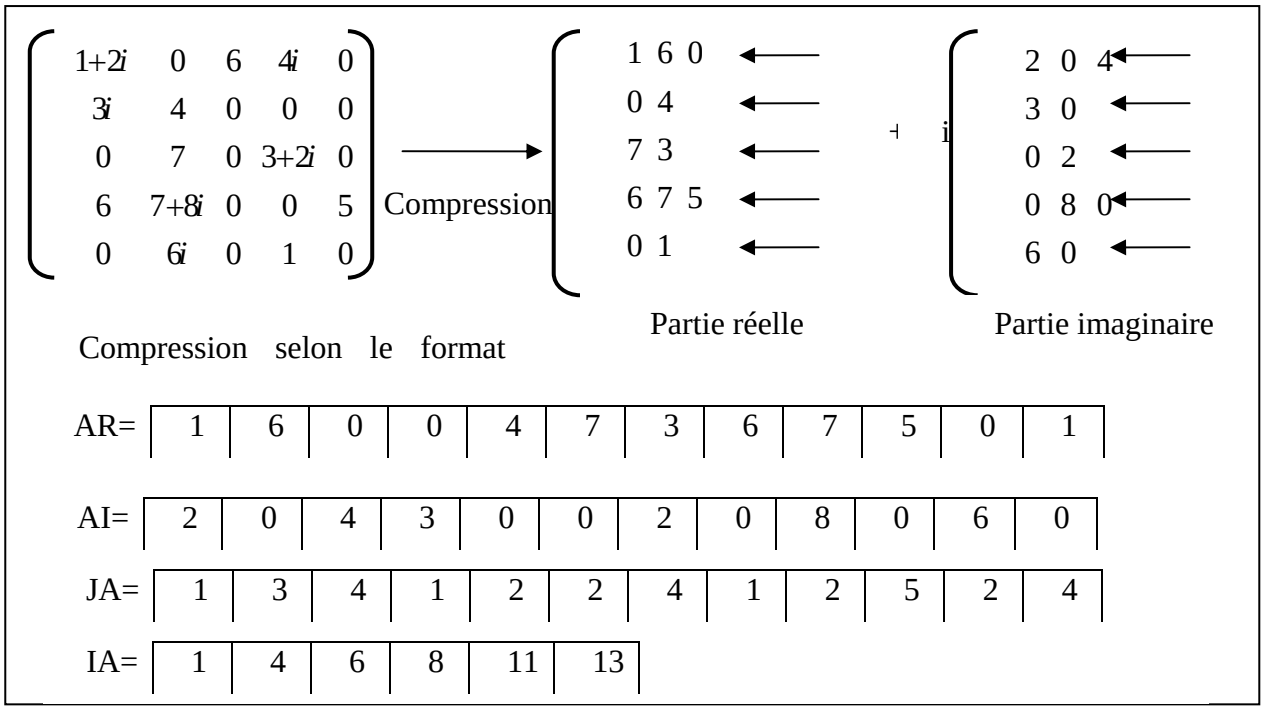


FIG 5.1- Exemple de compression d'une matrice complexe creuse au format CSR (compression et concaténation des lignes)

L'accès à l'élément M (*l*, *c*) de la matrice M se fera donc de la même façon dans la section 3.1.2 :

- Recherche des indices de début et de fin de la ligne n°*l* :

$$\begin{cases} \text{début=IA(l)} \\ \\ \text{fin= A(l+1)-1} \end{cases}$$

- Recherche de la valeur *c* dans **JA** entre **JA** (*début*) et **JA** (*fin*) (Cette recherche peut être obtenue en complexité logarithmique).
- Si *c* est trouvé à la position **JA**(*i*), la partie réelle de l'élément cherché est **AR**(*i*), la partie imaginaire de l'élément cherché est **AI**(*i*) ; si *c* n'est pas trouvé, l'élément cherché est 0.

Ce format a l'avantage de stocker les éléments dans des espaces de mémoire contigus et, ainsi, d'accélérer les accès à la mémoire, puisque plusieurs éléments contigus sont souvent rapatriés en même temps. Quant à la programmation du produit matrice-vecteur avec ce format, elle est assez simple : il suffit de parcourir une fois les tableaux **AR**, **AI** et **JA** guidé en cela par les valeurs de **IA**.

5.2.2 Format Ellpack-Itpack

Le format Ellpack-Itpack procède aussi à une compression des lignes. Pour les matrices complexes, Il utilise trois tableaux de dimensions (1 : *n*, 1 : LMAX), où *n* est le nombre de lignes de la matrice d'origine, et LMAX le nombre maximum d'éléments non nuls dans une ligne de cette matrice (voir l'exemple de la FIG 5.2) :

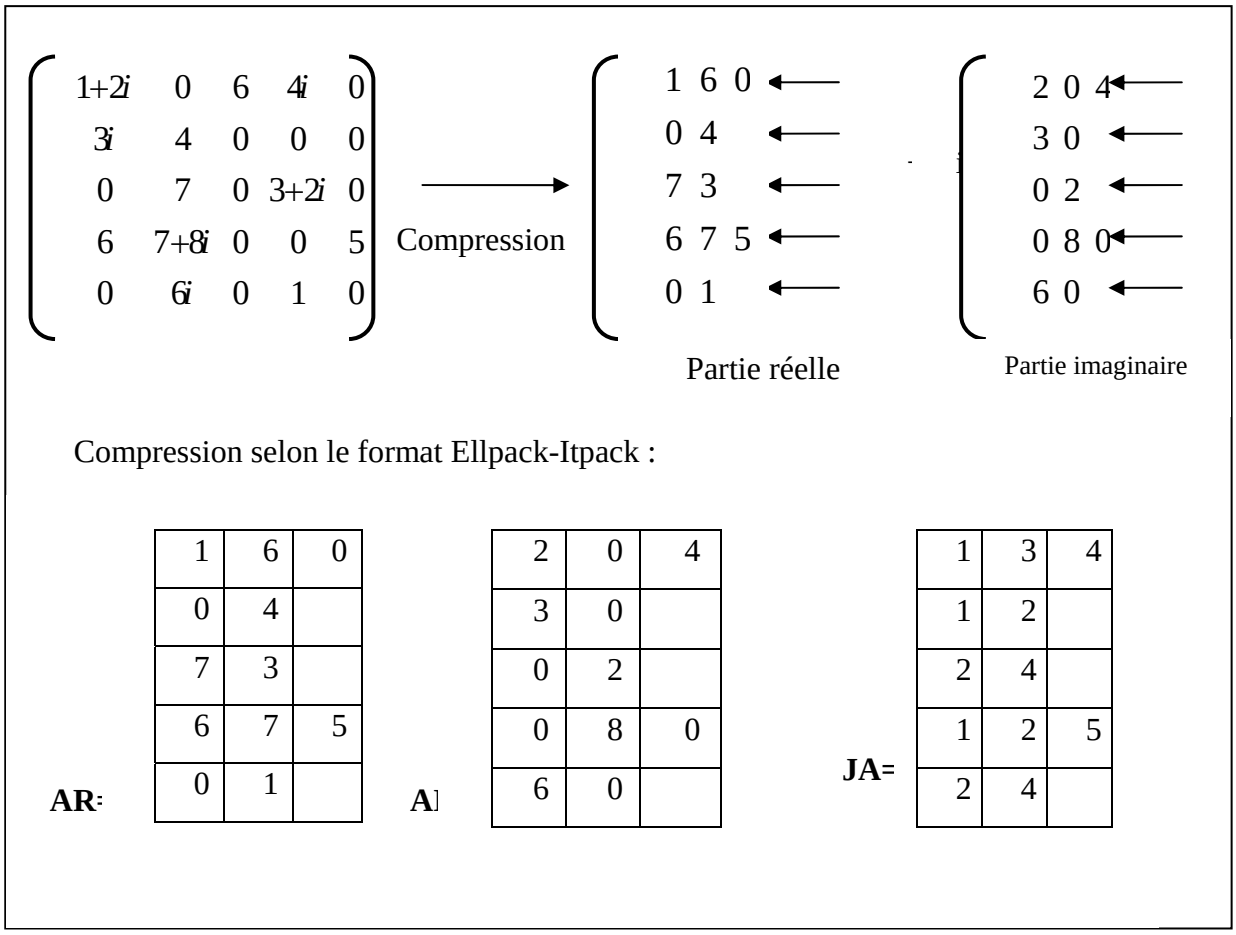


FIG 5.2- Exemple de compression d'une matrice complexe creuse au format Ellpack-Itpack (compression des lignes)

AR est un tableau dont chaque ligne contient les parties réelles des éléments non nuls d'une ligne de la matrice.

AI est un tableau dont chaque ligne contient les parties imaginaires des éléments non nuls d'une ligne de la matrice.
Comme pour le format **CSR**, on met 0 dans le tableau **AR** (respectivement **AI**) si la partie réelle (respectivement la partie imaginaire) d'un éléments non nul est nulle.

JA est un tableau dont chaque élément contient le numéro de colonne de l'élément correspondant de **AR** (et **AI**).

L'accès à l'élément **M** (*l*, *c*) de la matrice **M**, légèrement plus rapide que dans le format précédent, se fera de la façon suivante :

Recherche de la valeur *c* dans la ligne **JA** (*l*) (Cette recherche peut être obtenue en complexité logarithmique).

Si *c* est trouvé à la position **JA**(*i*), la partie réelle de l'élément cherché est **AR**(*i*), la partie imaginaire de l'élément cherché est **AI**(*i*) ; si c n'est pas trouvé, l'élément cherché est 0.

5.3 Les méthode hybride GMRES(m)/LS-Arnoldi dans le cas complexe

Dans l'espace complexe, **A** représente une matrice complexe, ***b*** le vecteur complexe second membre, et ***x*** le vecteur solution. D'après les transformations décrites à la section 5.1, nous faisons évoluer la méthode GMRES (m) pour variables complexes (Voir la FIG 5.3 : algorithme n°5*)*.

ALOGRITHME 5 : GMRES (m) pour variables complexes

1. Choisir la solution initiale x_0 ,

2. Calculer
$$r_0R = bR - (AR \cdot x_0R - AI \cdot x_0I)$$
$$r_0I = bI - (AI \cdot x_0R + AR \cdot x_0I) \quad ;$$
$$\|r_0\|_2 = \sqrt{(r_0R)^2 + (r_0I)^2} \quad , \text{ et} \quad v_1R = \frac{r_0R}{\|r_0\|_2}, \quad v_1I = \frac{r_0I}{\|r_0\|_2}$$

3. Répéter pour $j = 1,...,m$

$$h_{i,j}R = (AR \cdot v_jR - AI \cdot v_jI) \cdot v_iR + (AI \cdot v_jR + AR \cdot v_jI) \cdot v_iI$$
$$h_{i,j}I = (AI \cdot v_jI - AR \cdot v_jR) \cdot v_iI + (AI \cdot v_jR + AR \cdot v_jI) \cdot v_iR, (i = 1 \cdots j)$$

$$\hat{v}_{j+1}R = (AR \cdot v_jR - AI \cdot v_jI) - \sum_{i=1}^{j} (h_{i,j}R \cdot v_iR + h_{i,j}I \cdot v_iI)$$
$$\hat{v}_{j+1}I = (AI \cdot v_jR + AR \cdot v_jI) - \sum_{i=1}^{j} (-h_{i,j}R \cdot v_iI + h_{i,j}I \cdot v_iR)$$
$$h_{j+1,j} = \|\hat{v}_{j+1}\|_2 = \sqrt{(\hat{v}_{j+1}R)^2 + (\hat{v}_{j+1}I)^2}$$
$$v_{j+1}R = \frac{\hat{v}_{j+1}R}{h_{j+1,j}}$$
$$v_{j+1}I = \frac{\hat{v}_{j+1}I}{h_{j+1,j}}$$

4. Etablir la solution approchée :
$$x_mR = x_0R + (V_mR \cdot y_mR - V_mI \cdot y_mI)$$
$$x_mR = x_0R + (V_mI \cdot y_mR + V_mR \cdot y_mI),$$
où y_m minimise $\|\beta e_1 - \overline{H}_m y\|_2$, $y \in \Re^m$,

5. Redémarrage :

Calculer
$$r_mR = bR - (AR \cdot x_mR - AI \cdot x_mI)$$
$$r_mI = bI - (AI \cdot x_mR + AR \cdot x_mI)$$

Si r_m est satisfaisant alors arrêter,

Sinon réinitialiser $x_0R = x_mR$, $r_0R = r_mR$, $x_0I = x_mI$, $r_0I = r_mI$,

$$\beta = \|r_0\|_2 = \sqrt{(r_0R)^2 + (r_0I)^2} \quad \text{et} \quad v_1R = \frac{r_0R}{\beta}, \quad v_1I = \frac{r_0I}{\beta}$$

retourner à l'étape n°2

FIG 5.3-Algorithme n° 5 : La méthode GMRES (m) pour variables complexes

Pour le cas complexe, la méthode hybride GMRES/LS-Arnoldi (Voir la FIG 5.4 : algorithme n°6) utilise des paramètres identiques comme la section 3.2.3 (liste non exhaustive). Ceux-ci ont une grande influence sur les performances obtenues par ces méthodes numériques, spécialement par la méthode hybride :

n est la taille de la matrice ***A.***

m est la taille du sous-espace de Krylov. On l'utilise non seulement dans la méthode GMRES qui donne la solution du système linéaire mais aussi dans la méthode d'Arnoldi qui calcule les valeurs propres. Quand on fait décroître la valeur de *m*, pour chaque itération la durée diminue, néanmoins le nombre d'itérations nécessaires à la convergence augmente. On pourra mettre en évidence une valeur optimale de ce paramètre. Pour ces deux algorithmes, elle peut être différente. En outre, on peut profiter de ces deux valeurs de *m* pour régler la vitesse relative des deux calculs, afin d'ajuster le nombre d'envois de valeurs propres pour une efficacité optimale. Il faudra donc qu'on distingue bien *m***(GMRES)** de *m***(Arnoldi)**.

k est le degré du polynôme « Least Squares »

l est la puissance du calcul « Least Squares ». Ce paramètre est le nombre d'itérations « Least Squares » exécutées à chaque prise en compte d'une nouvelle série de valeurs propres supplémentaires.

ALGORITHME 6 : GMRES(*m*)/LS(*k*,*l*)-Arnoldi(m2) [Partie GMRES/LS] dans le cas complexe

1. Choisir le vecteur initial x_0 , la dimension *m* du sous-espace de Krylov pour GMRES, *k* le degré du polynôme Least Squares, *l* le nombre d'itérations Least Squares successives.
2. Calculer x_m , le $m^{ième}$ itéré de GMRES démarré avec x_0 .

$$x_m R = x_0 R + (V_m R \cdot y_m R - V_m I \cdot y_m I)$$
$$x_m R = x_0 R + (V_m I \cdot y_m R + V_m R \cdot y_m I),$$

Faire $x_0 = x_m$ et $r_0 = b - Ax_0$

$$r_0 R = bR - (AR \cdot x_0 R - AI \cdot x_0 I)$$
$$r_0 I = bI - (AI \cdot x_0 R + AR \cdot x_0 I)$$
$$\|r_0\|_2 = \sqrt{(r_0 R)^2 + (r_0 I)^2}$$

Si $\|r_0\|_2 < \varepsilon_g$ alors arrêter

3. Si un nouveau polynôme P_k est disponible,
   alors faire l'étape n°4 (Least Squares)
   Sinon
   aller à l'étape n°2(GMRES)
   fin si
4. Pour $i = 1, \cdots, l$
   calculer $\tilde{x} = x_0 + P_k(A) r_0$
   faire $x_0 = \tilde{x}$ et $r_0 = b - Ax_0$
   fin pour
   Si $\|r_0\|_2 < \varepsilon_g$ alors
   arrêter
   Sinon
   aller à l'étape n°2 (GMRES)

FIG 5.4-Algorithme n° 6 : La méthode GMRES (m)/LS-Arnoldi pour variables complexes

5.4 Les matrices « young1c » et « dwg961b »

Des matrices industrielles, issues du serveur Web « MatrixMarket » ont été utilisées pour vérifier la méthode de GMRES (m) pour variables complexes:

- « young1c », est une matrice complexe non- Hermitienne ;

« dwg9601b », est une matrice complexe Hermitienne ;

Afin de vérifier l'exactitude des valeurs numériques des solutions trouvées par l'algorithme, le vecteur second membre *b* a été choisi tel que la solution du système soit $x = (\frac{1}{n}, \frac{1}{n}, ..., \frac{1}{n})^T$ (on a alors $b = Ax$). Les méthodes démarrent avec $x_0 = (0, 0, ..., 0)^T$.

Ces matrices ont 4089 éléments non nuls et proviennent d'une application en acoustique : « young1c » est une matrice complexe non Hermitienne (Voir *FIG 5.5* et *FIG 5.6*).

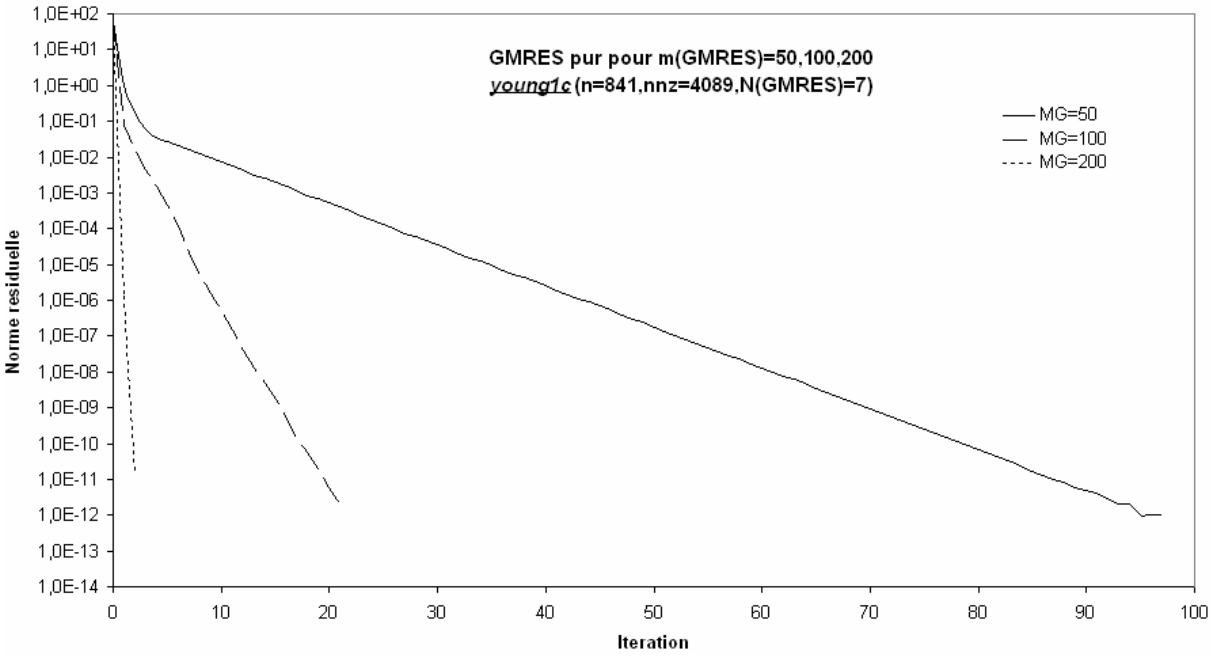


FIG 5.5-Evolution de la norme résiduelle sur l'itération pour la matrice « young1c »

Cette figure FIG 5.5 décrit le nombre d'itérations nécessaires à la méthode traditionnelle GMRES(***m***) pour la matrice complexe «*young1c*» en fonction de ***MG***, ***MG*** s'accroît de 50 à 200, le nombre d'itération décroît . Quand ***MG*** est 200, on a la moindre itération à atteindre la convergence.

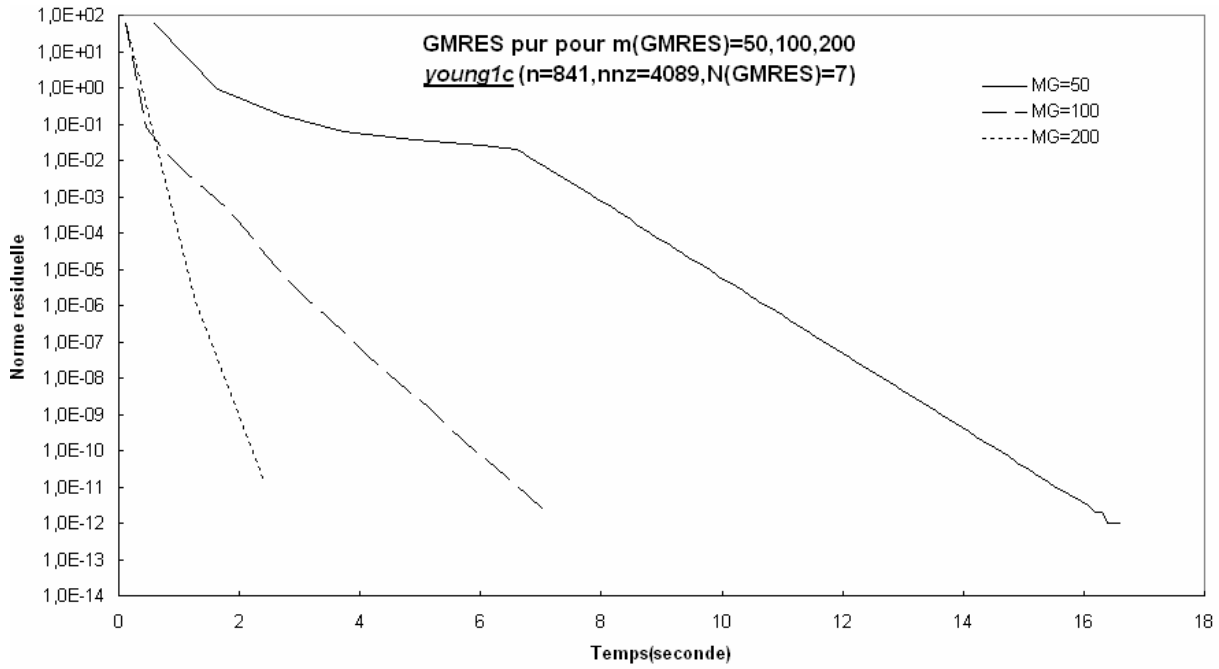


FIG 5.6-Evolution de la norme résiduelle sur le temps pour la matrice « young1c »

Cette figure FIG 5.6 décrit le temps nécessaire à la méthode traditionnelle GMRES(***m***) pour la matrice complexe «*young1c*» en fonction de ***MG***, ***MG*** s'accroît de 50 à 200, le temps permettant d'atteindre la convergence devient de plus en plus court . Quand ***MG*** est 200, le temps d'atteindre à la convergence est le plus court.

« dwg9601b » (taille 961x961, 10591 éléments non nuls) est une matrice complexe symétrique, provenant d'une application de guide d'ondes en calcul de structures.

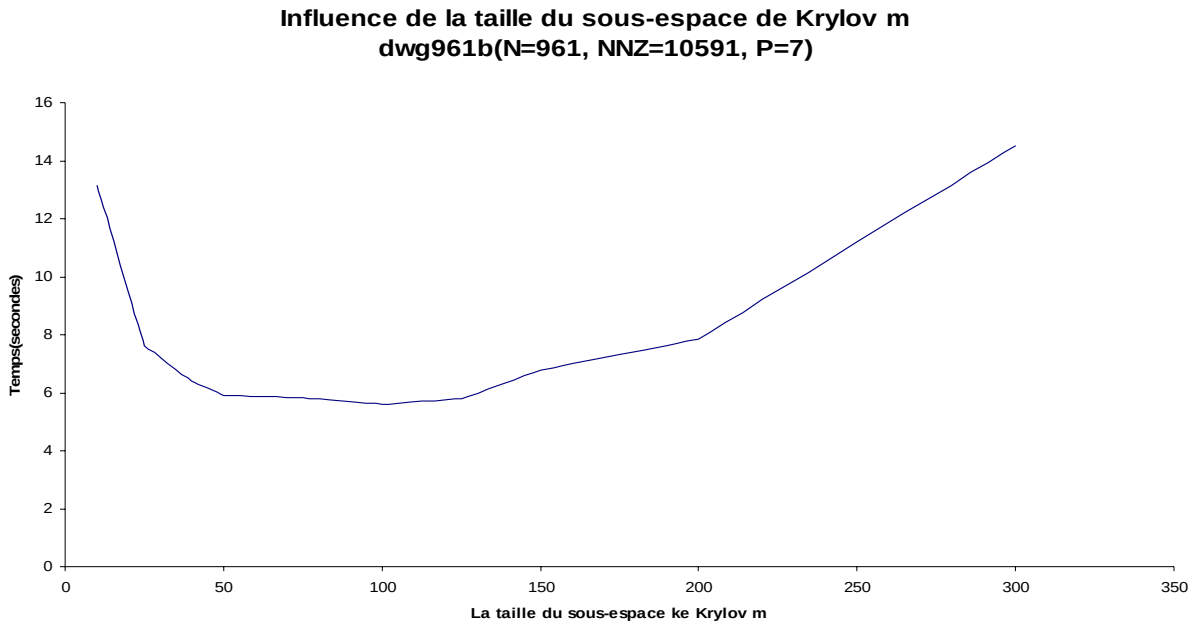


FIG 5.7-L'influence en fonction de la taille du sous-espace de Krylov m pour GMRES(m) pur (matrice « dwg961b »)

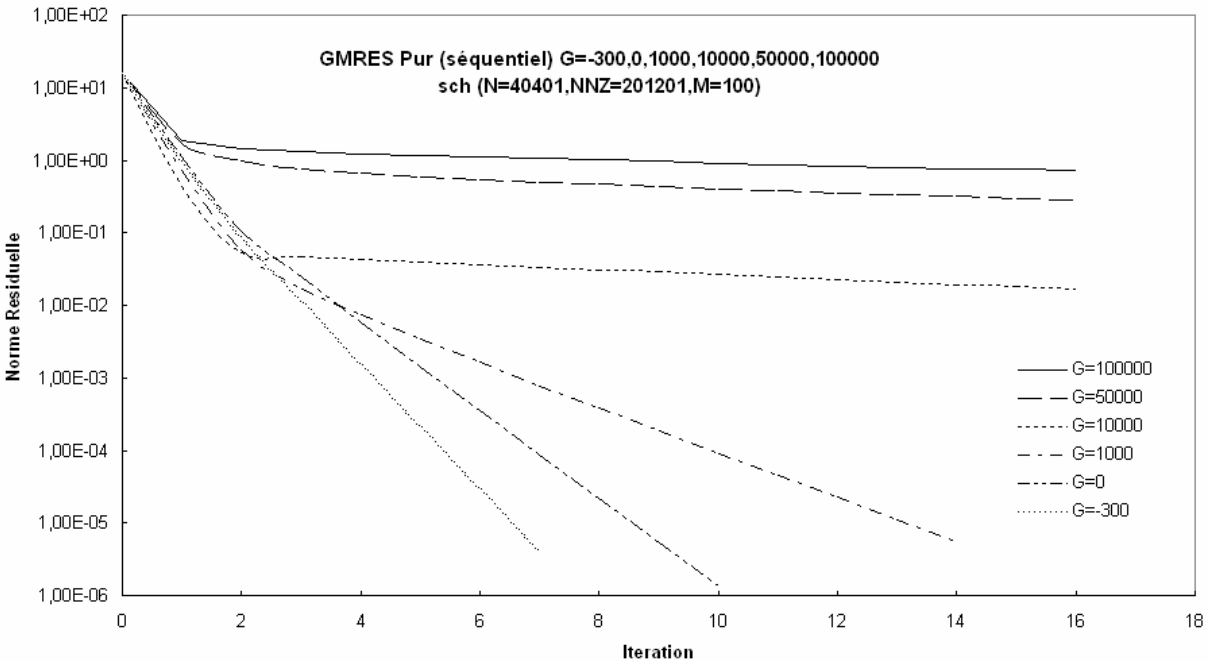
Nous l'utilisons pour présenter l'influence du paramètre *m*, qui est la taille du sous-espace de Krylov, sur la méthode GMRES(***m***) pour les matrices à variables complexes(Voir FIG 5.7).

La FIG 5.7 présente les temps nécessaires pour atteindre la convergence demandée en fonction du paramètre *m*. L'augmentation de *m* diminue le nombre d'itérations nécessaires à la convergence, mais augmente la durée de chaque itération. On comprend que dans ces conditions *m* ne puisse pas dépasser un certain seuil, au delà duquel le temps nécessaire à la convergence augmente.

La valeur de *m* devra donc être choisie pour obtenir une efficacité optimale. Mais ce choix n'est pas simple, car il dépend de la matrice utilisée dont il n'est pas facile, a priori de prévoir les meilleurs paramètres. La valeur de *m* utilisée au cours des tests a été déterminée empiriquement.

5.5 Les matrices « SCH » du CEA, avec la méthode GMRES pure

Les matrices « SCH » du CEA ciblées sont de taille N=40401, avec NNZ=201201 éléments non nuls. La difficulté de convergence augmente avec le paramètre « G » (gamma dans le rapport du CEA [32]) qui est un paramètre important de l'application et donc du programme construisant la matrice « SCH ». Ce programme nous a été fourni par le CEA dans le cadre d'un contrat. L'influence du paramètre « G » est confirmée expérimentalement. En effet, on peut la remarquer dans la *FIG 5.8*. Le nombre des itérations et le temps utilisé augmentent manifestement avec « G ».



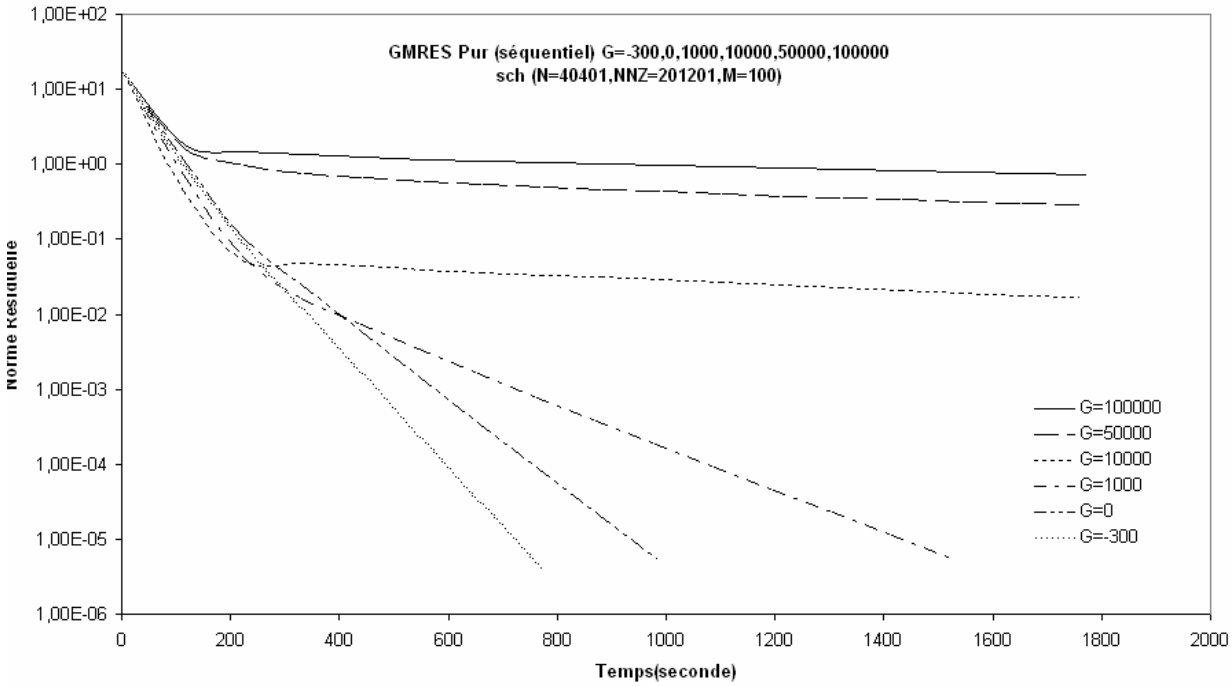


FIG 5.8 -L'influence du paramètre « G » (matrice «sch»)

Dans cette figure FIG 5.8, le temps permettant d'atteindre la convergence de la méthode traditionnelle GMRES(***m***) qui fonctionne séquentiellement sur un seul processeur pour la matrice «SCH» diminue avec l'augmentation de « ***G*** » . Quand ***G*** s'accroît de -300 à 100000, la convergence devient de plus en plus difficile.

Après cette première tentative, nous avons fait de nombreux tests. Dans les figures *FIG 5.9*-*FIG 5.14*, on vérifie que l'influence du paramètre « M » (la taille de sous-espace de Krylov) de la méthode « GMRES » est significative. Quand on augmente la taille du sous-espace le nombre d'itérations et le temps du calcul diminuent clairement comme prévu.

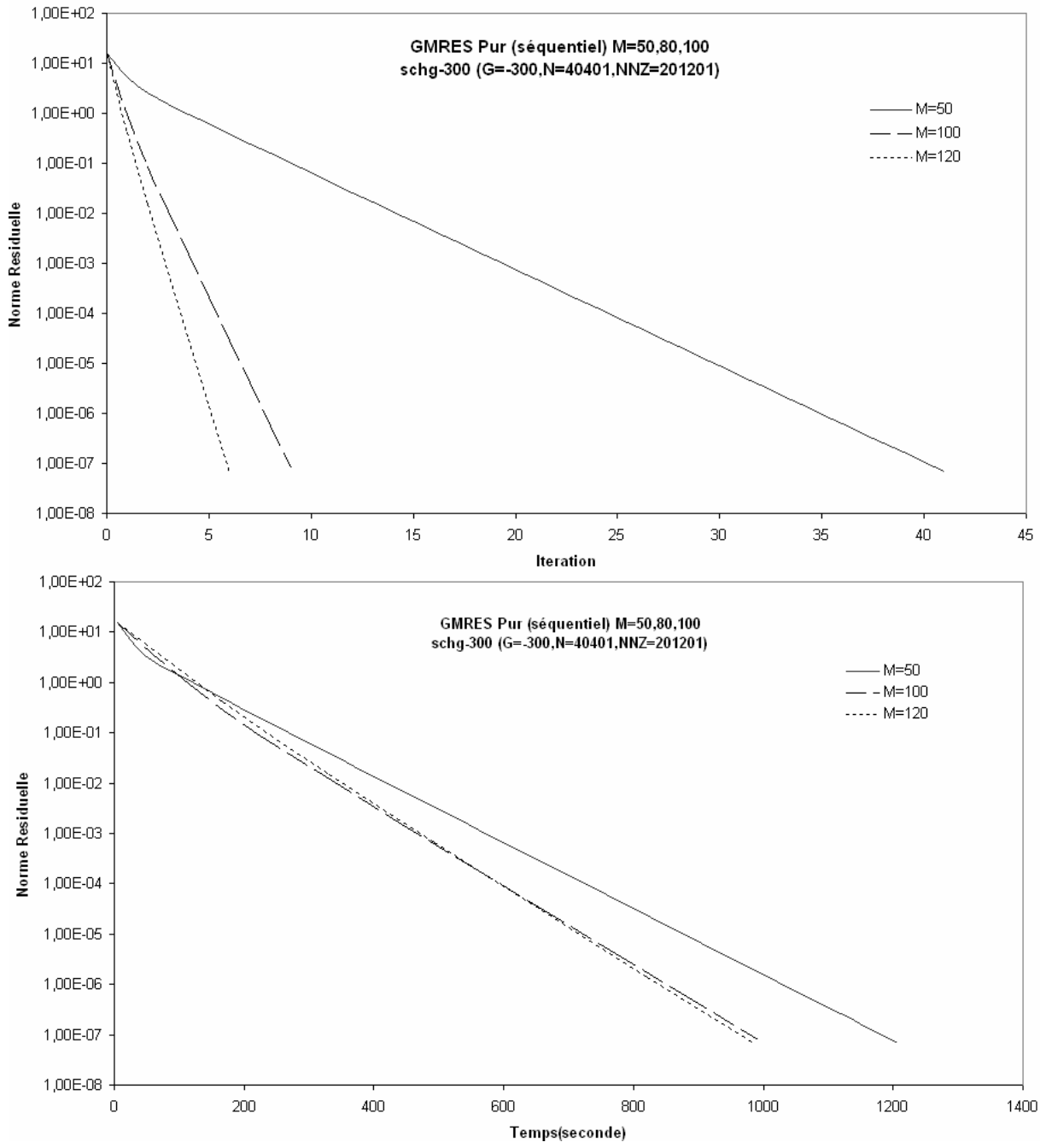


FIG 5.9-L'influence du paramètre « M » (matrice «schg-300»)

Dans cette figure FIG 5.9, le temps permettant d'atteindre la convergence de la méthode traditionnelle GMRES(***m***) qui fonctionne séquentiellement sur un seul processeur pour la matrice «SCH» diminue avec l'augmentation de ***M*** quand le paramètre «***G*** » est « -300 » . Quand ***M*** s'accroît de 50 à 120, le temps pour atteindre la convergence devient de plus en plus court.

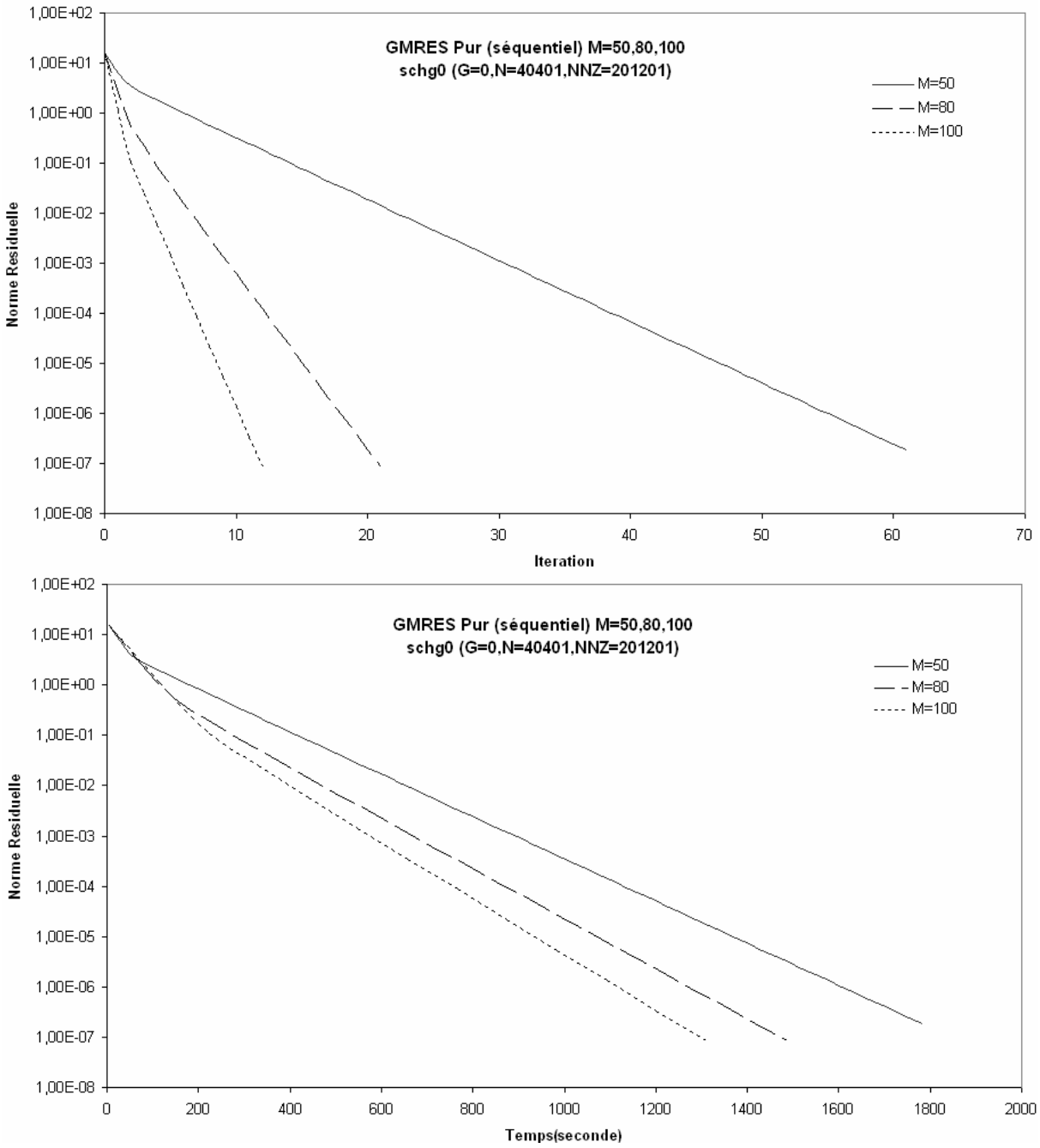


FIG 5.10-L'influence du paramètre « M » (matrice «schg0»)

Dans cette figure FIG 5.10, le temps permettant d'atteindre la convergence de la méthode traditionnelle GMRES(***m***) qui fonctionne séquentiellement sur un seul processeur pour la matrice «SCH» diminue avec l'augmentation de ***M*** quand le paramètre «***G*** » est « 0 » . Quand ***M*** s'accroît de 50 à 100, le temps pour atteindre la convergence devient de plus en plus court.

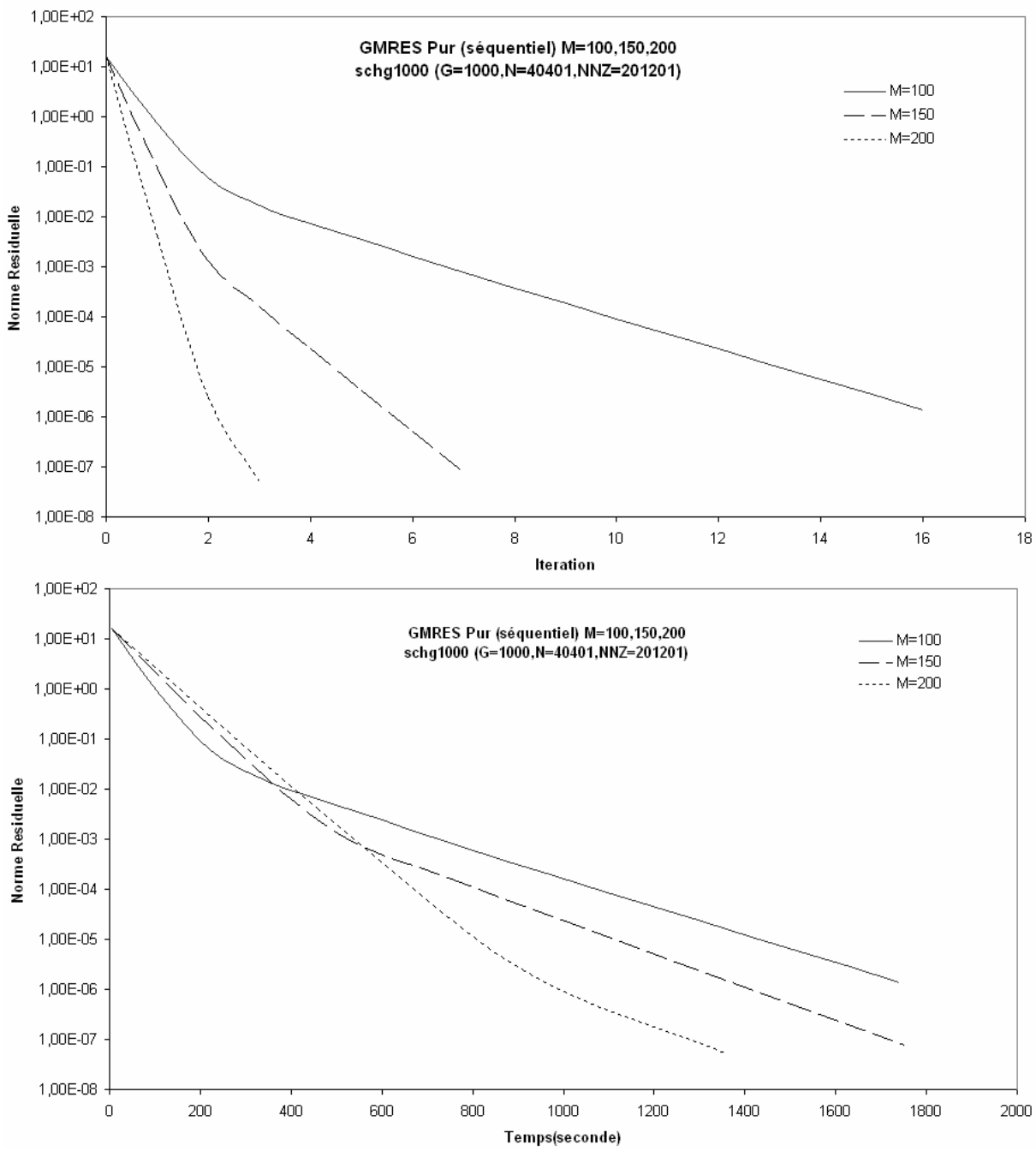


FIG 5.11-L'influence du paramètre « M » (matrice «schg1000»)

Dans cette figure FIG 5.11, le temps permettant d'atteindre la convergence de la méthode traditionnelle GMRES(***m***) qui fonctionne séquentiellement sur un seul processeur pour la matrice «SCH» diminue avec l'augmentation de ***M*** quand le paramètre «***G*** » est « 1000 » . Quand ***M*** s'accroît de 100 à 200, la convergence est plus rapidement atteinte.

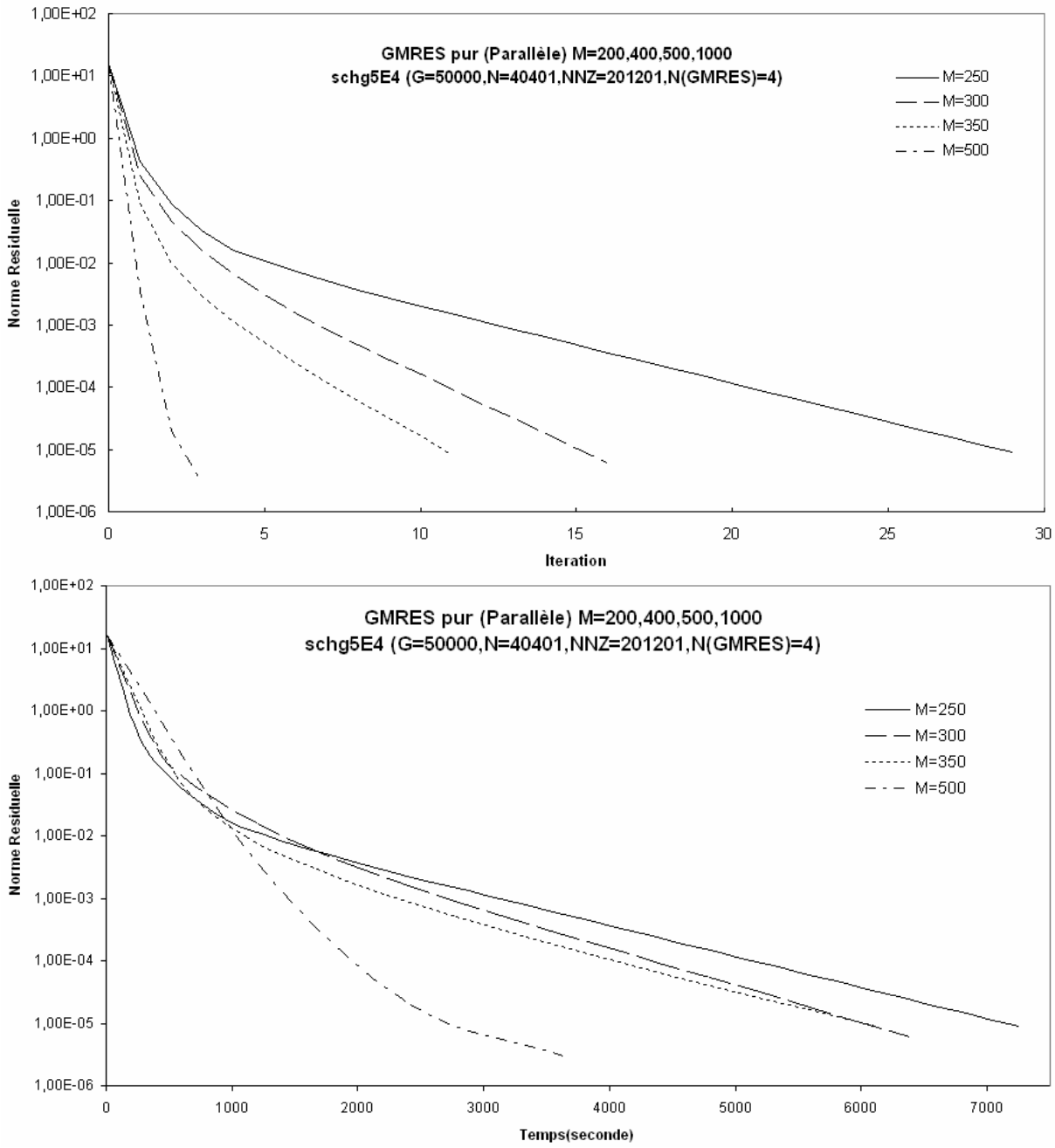


FIG 5.12-L'influence du paramètre « M » (matrice «schg5E4»)

Dans cette figure FIG 5.12, le temps permettant d'atteindre la convergence de la méthode traditionnelle GMRES(***m***) qui fonctionne parallèlement sur 4 processeurs pour la matrice «SCH» diminue également avec l'augmentation de ***M*** quand le paramètre «***G*** » est « 50000 » . Quand ***M*** s'accroît de 250 à 500, la convergence est de plus en plus rapidement atteinte.

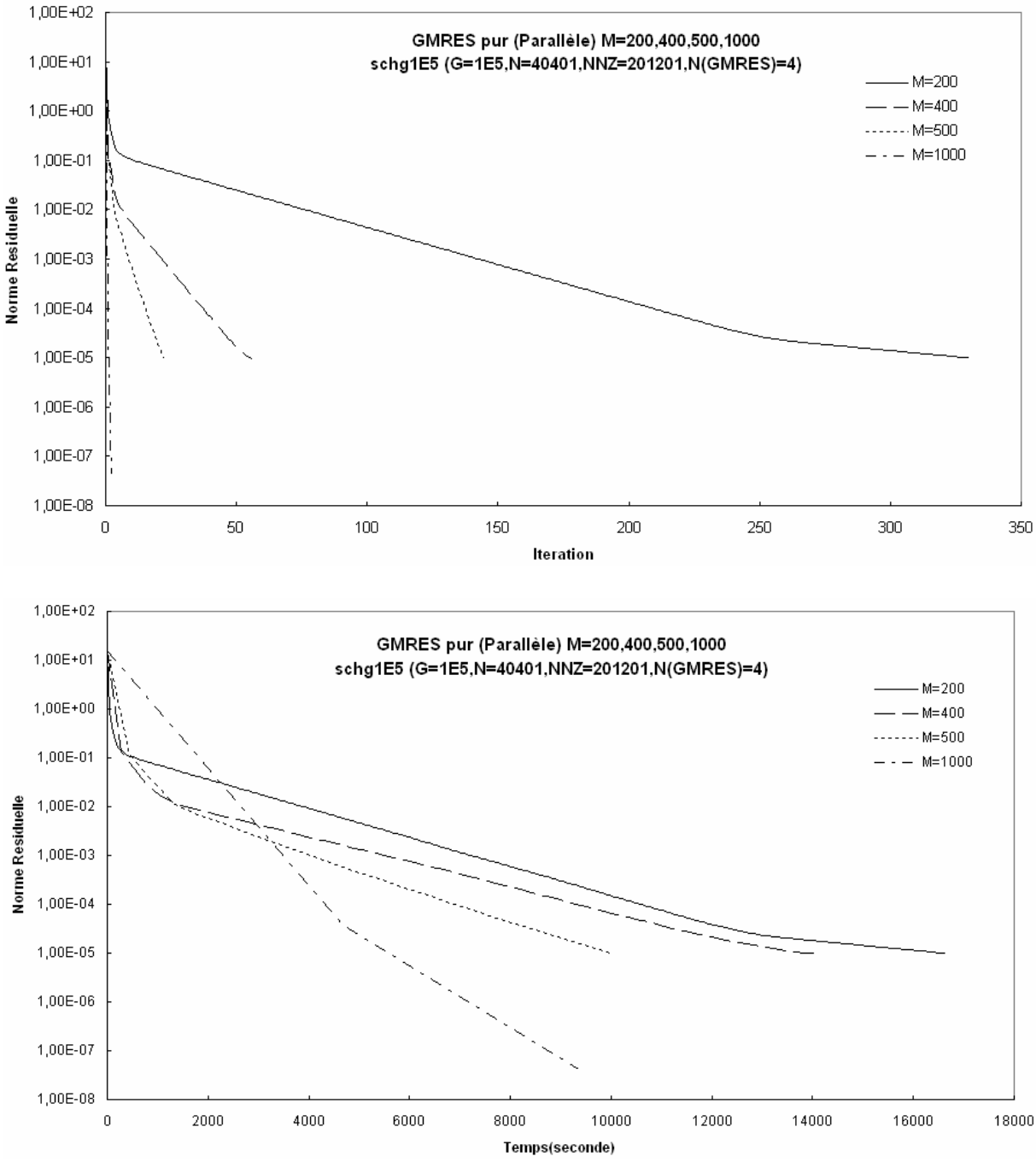


FIG 5.13-L'influence du paramètre « M » (matrice «schg1E5»)

Dans cette figure FIG 5.13, le temps permettant d'atteindre la convergence de la méthode traditionnelle GMRES(***m***) qui fonctionne parallèlement sur 4 processeurs pour la matrice «SCH» diminue également avec l'augmentation de ***M*** quand le paramètre «***G*** » est « 100000 » . Quand ***M*** s'accroît de 200 à 1000, la convergence est de plus en plus rapidement atteinte.

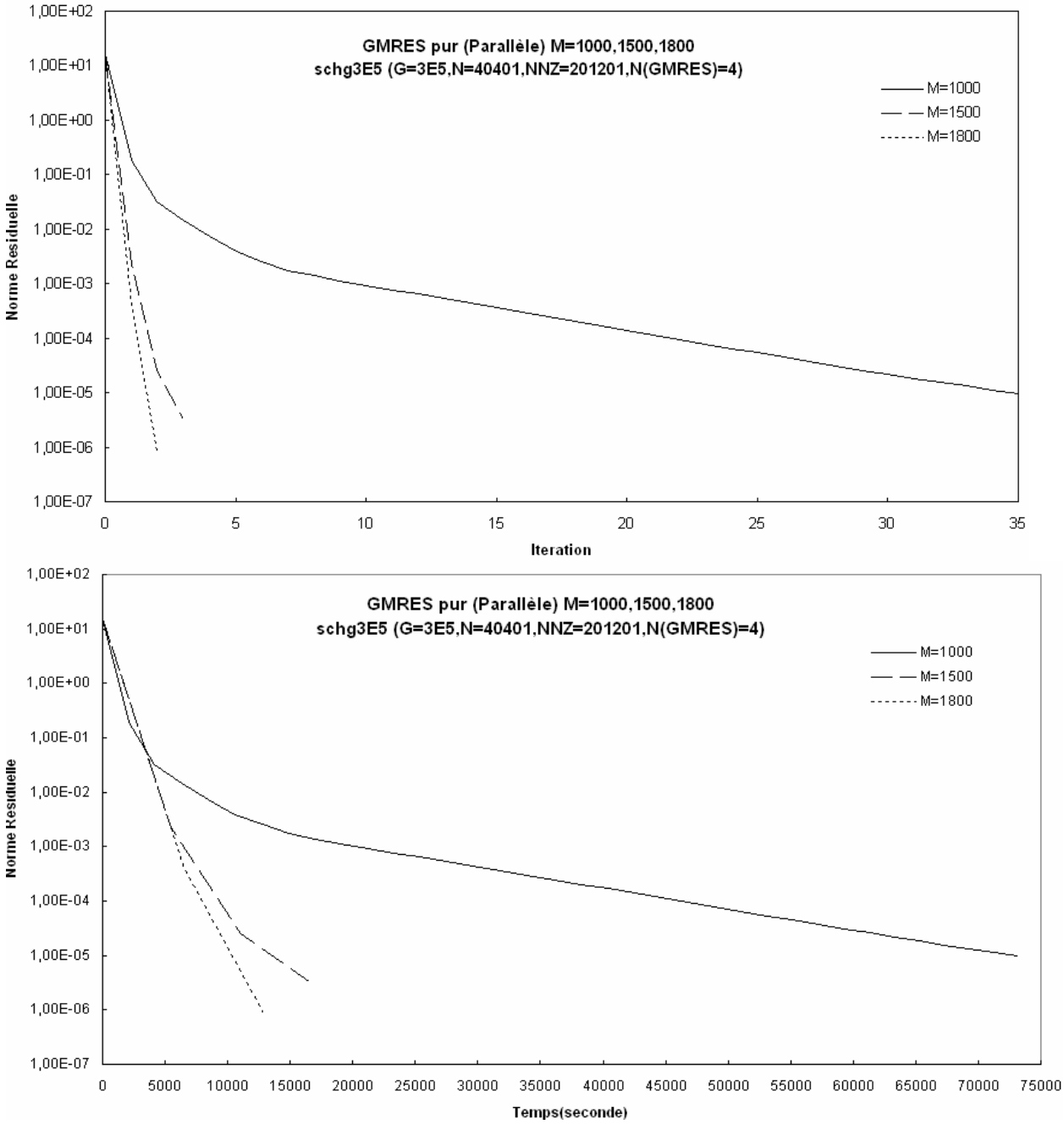


FIG 5.14-L'influence du paramètre « M » (matrice «schg3E5»)

Dans cette figure FIG 5.14, le temps permettant d'atteindre la convergence de la méthode traditionnelle GMRES(***m***) qui fonctionne parallèlement sur 4 processeurs pour la matrice «SCH» diminue également avec l'augmentation de ***M*** quand le paramètre «***G*** » est « 300000 » . Quand ***M*** s'accroît de 1000 à 1800, la convergence est de plus en plus rapidement atteinte.

5.6 La méthodes hybrides pour les matrices « SCH » du CEA

Nous rappelons que les matrices « SCH » sont de taille N=40401, avec NNZ=201201. La difficulté de la convergence augmente avec le paramètre « G ».

Mathématiquement, la méthode hybride n'est pas valide telle que nous l'avons présentée et testée dans le chapitre précédent. En effet, le produit scalaire entre polynômes utilisé s'appuie sur la propriété que les valeurs propres d'une matrice réelle sont conjuguées. Ceci n'est plus vrai dans le cas d'une matrice à variables complexes. Nous introduisons donc une première heuristique très grossière. Nous allons systématiquement prendre le conjugué des valeurs propres approximées à l'aide de la méthode d'Arnoldi. Nous pouvons ensuite calculer l'ellipse et effectuer la minimisation comme dans le cas réel. Il s'agit d'une heuristique grossière que nous avons voulu tester avant de regarder en profondeur le coté mathématique, très difficile. Il est aussi envisageable de limiter cette technique aux valeurs propres uniquement à une certaine distance de l'axe des réels, afin de ne pas agrandir de trop la partie de l'espace complexe prise en compte pour calculer l'ellipse. Cette distance devient alors un nouveau paramètre de la méthode.

Nous avons utilisé la méthode hybride sur 8 processeurs. Un processeur s'occupe des communications, 4 processeurs sont attribués à l'exécution de GMRES(m), 2 processeurs sont alloués pour le paquetage « PARPACK » qui calcule les valeurs propres et 1 processeur est pour la méthode « Least Square ». Pour la version parallèle de « GMRES » seul, un processeur se charge de la communication, 4 processeurs sont attribués pour faire GMRES(m).

Nous remarquons, cf. les figures *FIG 5.15*,*FIG 5.16*,*FIG 5.17*, que dans les cas de convergence facile pour GMRES seul (qualifié de « pur »), la méthode hybride n'est pas efficace. Dans le cas complexe, les résidus sont presque toujours égaux après la prise en compte de la méthode « LS-Arnoldi ». On peut remarquer que les hauteurs des pics sont presque les mêmes. La méthode « LS-Arnoldi » ne parvient pas à bien optimiser le vecteur de redémarrage. En fait, plutôt que d'accélérer la convergence, cela revient à prendre un nouveau vecteur initial, tellement l'heuristique est grossière. De même, nous avons fait des tests en prenant en compte la distance à l'axe réel des valeurs propres pour n'en

sélectionner qu'un sous-ensemble ; sans plus de succès. En fait, nous nous attendions à ces résultats mais il convenait de tester pour ne pas passer à coté d'une heuristique simple. Par ailleurs, cette heuristique n'a pas demandé beaucoup de temps de développement.

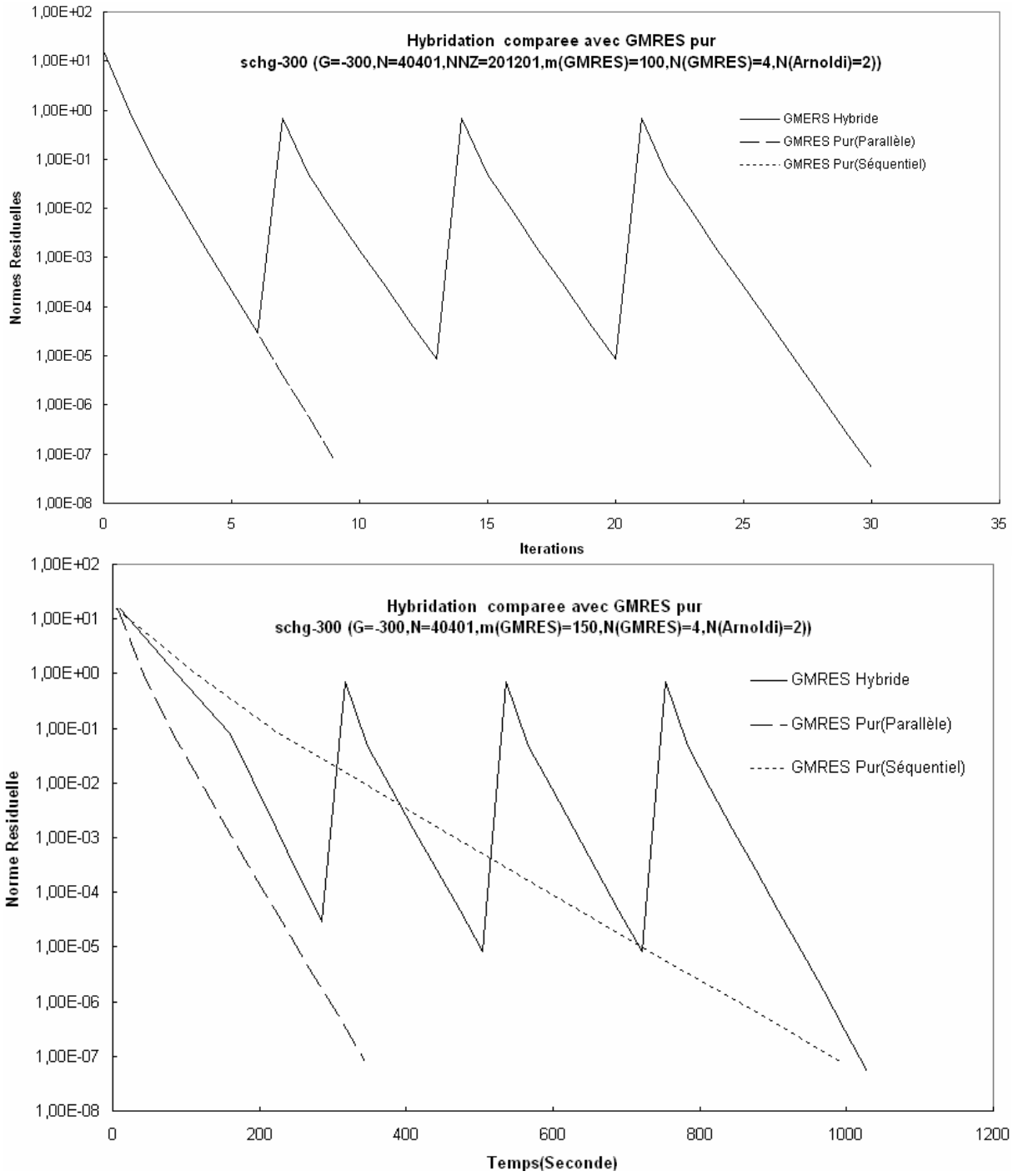


FIG 5.15 -Evolution de la norme résiduelle avec la méthode hybride comparée avec GMRES Pur (matrice schg-300)

Dans cette figure FIG 5.15, le temps permettant d'atteindre la convergence par la méthode hybride est plus long que par la méthode traditionnelle GMRES(***m***) qui

fonctionne parallèlement sur 4 processeurs, mais très proche de celui de la méthode traditionnelle GMRES(***m***) qui fonctionne séquentiellement sur un seul processeur pour la matrice «SCH» quand le paramètre «***G*** » est « -300 » . Les nombreux pics de prise en compte des valeurs propres par la méthode hybride semblent avoir ralenti sa convergence.

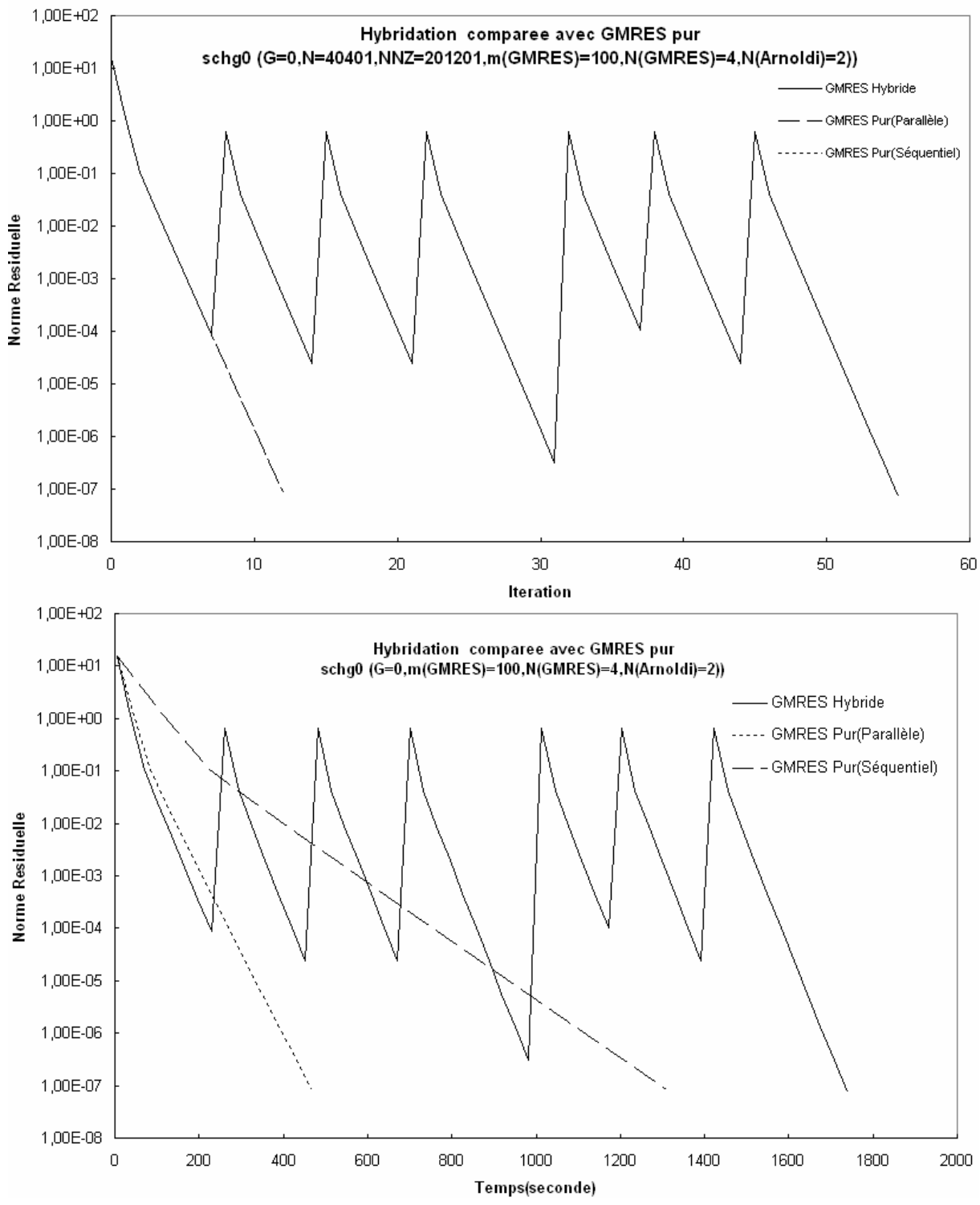
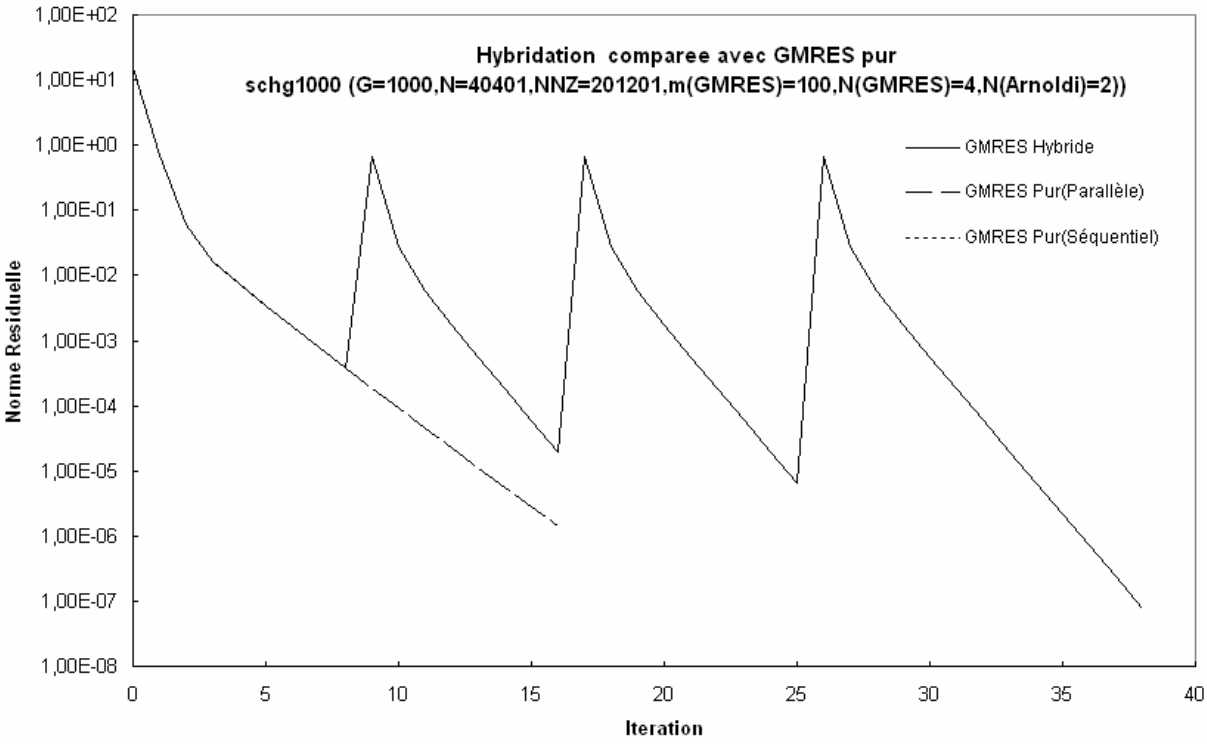


FIG 5.16 -Evolution de la norme résiduelle avec la méthode hybride comparée avec GMRES Pur (matrice schg0)

Dans cette figure FIG 5.16, le temps permettant d'atteindre la convergence par la méthode hybride est plus long que par la méthode traditionnelle GMRES(***m***) qui fonctionne parallèlement sur 4 processeurs, mais aussi plus long que par la méthode traditionnelle GMRES(***m***) qui fonctionne séquentiellement sur un seul processeur pour la matrice «SCH» quand le paramètre «***G*** » est « 0 » . Nous remarquons également les nombreux pics de la courbe « GMRES hybride » qui semblent avoir ralenti la convergence.



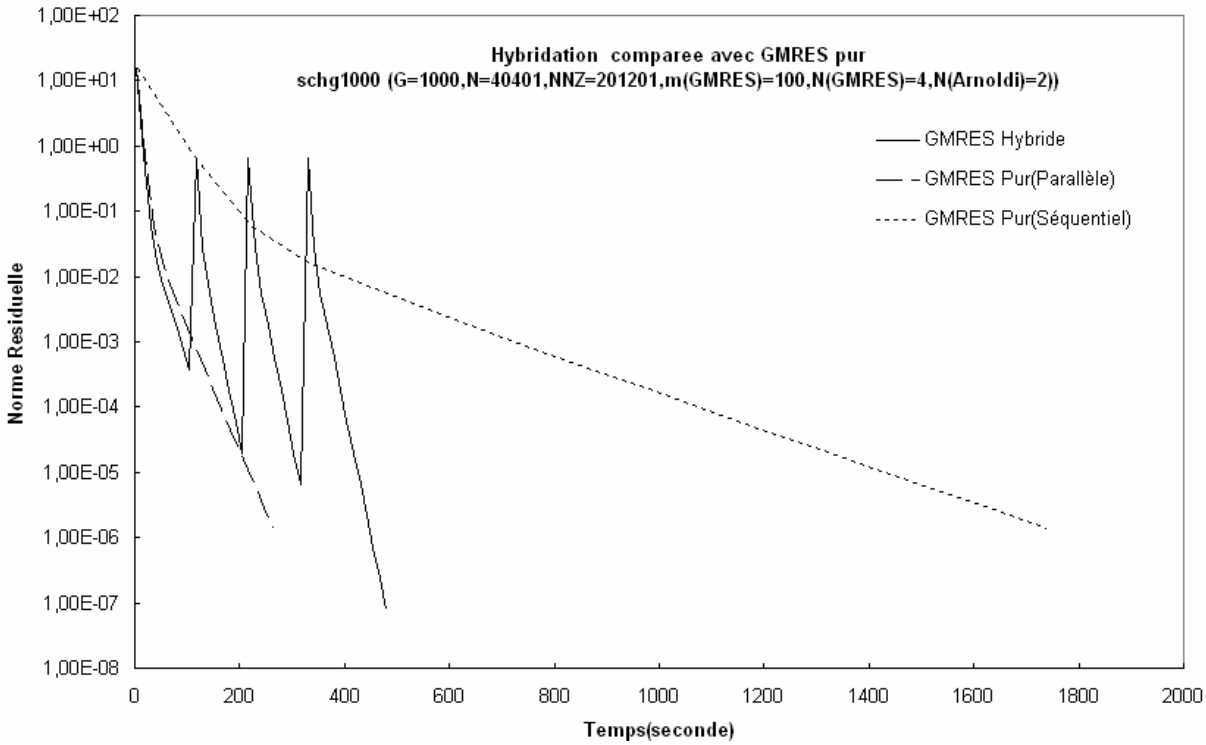


FIG 5.17 -Evolution de la norme résiduelle avec la méthode hybride comparée avec GMRES Pur (matrice schg1000)

Dans cette figure FIG 5.17, le temps permettant d'atteindre la convergence par la méthode hybride est plus long que par la méthode traditionnelle GMRES(***m***) qui fonctionne parallèlement sur 4 processeurs, mais beaucoup plus court que par la méthode traditionnelle GMRES(***m***) qui fonctionne séquentiellement sur un seul processeur pour la matrice «SCH» quand le paramètre «***G*** » est « 1000 » . Nous remarquons également que plusieurs pics de la courbe « GMRES hybride » semblent avoir ralentis la convergence par rapport à la méthode parallèle GMRES(***m***) pur.

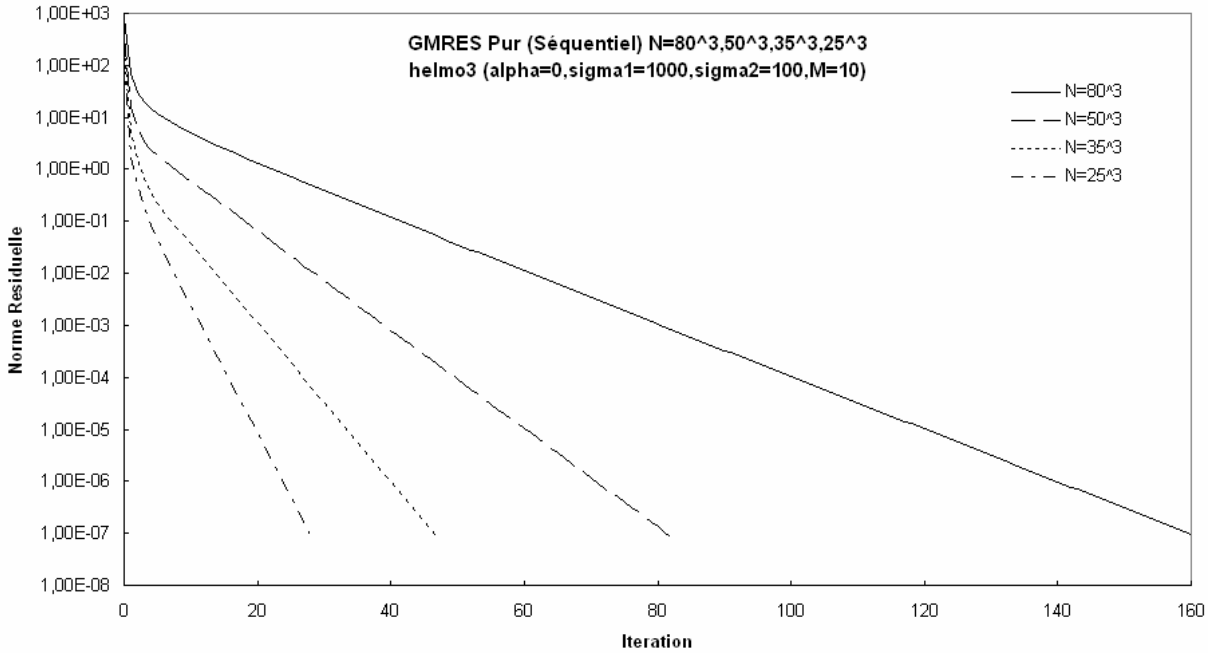
5.7 La méthode GMRES seule pour les matrices « HELMO » du CEA

Dans les *FIG 5.18*,FIG 5.21, on présente les tests effectués avec différentes tailles de matrice. Avec la même taille du sous-espace de Krylov, le nombre des itérations et le temps total de calcul augmentent évidemment avec la taille de la matrice.

Dans les figures FIG 5.19,FIG 5.20, on remarque, comme attendu, que le nombre des itérations diminue lorsque la taille du sous-espace Krylov croît, Il n'en est pas de même

pour le temps total de calcul qui, alors augmente. La courbe correspondant à la meilleure convergence est celle avec M=100. La deuxième est pour M=10. Comme déjà remarqué, avec l'augmentation de M, le nombre d'itérations diminue mais le temps de chaque itération augmente. C'est pourquoi la courbe « M=10 » est plus rapide. Quand à la courbe « M=100 », grâce à une taille assez grande, après seulement deux itérations, les résultats convergent. Bien que le temps de calcul de chaque itération soit long, le faible nombre d'itérations (deux itérations), donne un temps de calcul total plus court.

Dans les figures FIG 5.22,FIG 5.23 pour l'algorithme parallèle, on remarque comme prévu, que le nombre d'itérations diminue lorsque la taille du sous-espace de Krylov croît, mais il n'en est plus de même pour le temps total de calcul, qui parfois alors augmente. Pour la matrice avec la taille plus grande « helmo3N80 », le phénomène est plus évident. Dans la figure FIG 5.23, la courbe correspondant à la meilleure convergence est celle avec M=100, le moins d'itérations et aussi le moins de temps de calcul. Mais la deuxième courbe la plus rapide est celle avec M=10, pas celle avec M=75 comme attendu. Parce qu'avec l'augmentation de M, le nombre d'itérations diminue mais le temps de chaque itération augmente. La même explication s'applique aux cas de M=25, 50,75.



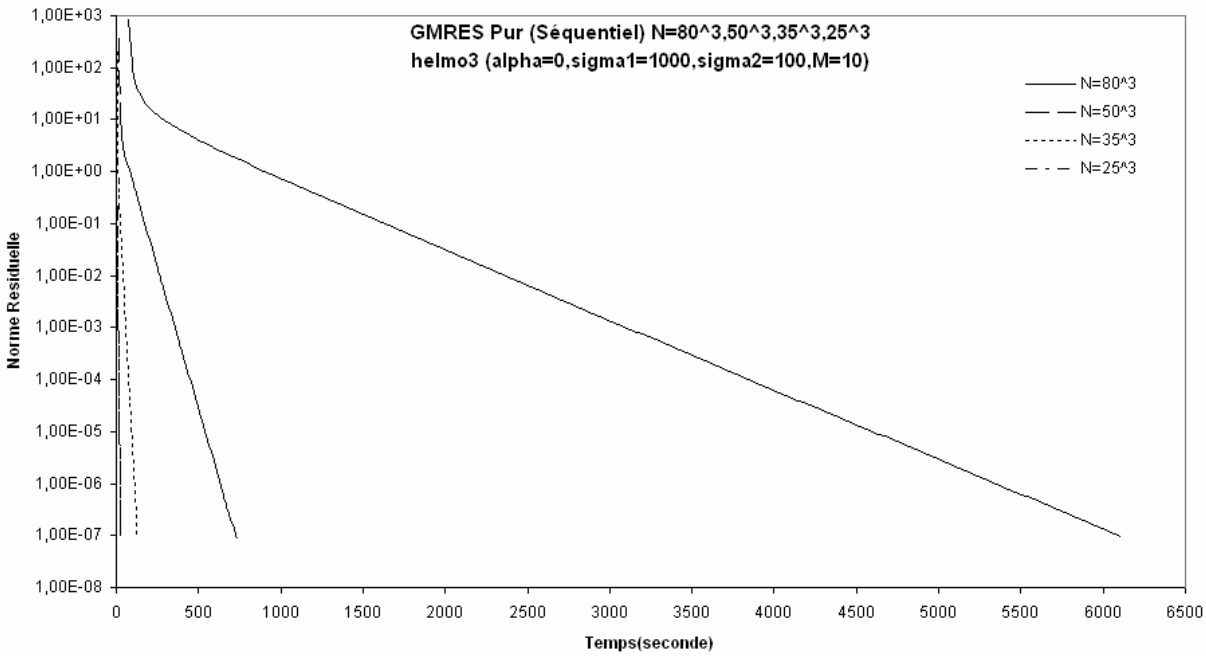


FIG 5.18 -L'influence de la taille N des matrices « helmo3 »

Dans cette figure FIG 5.18, le temps permettant d'atteindre la convergence de la méthode traditionnelle GMRES(***m***) qui fonctionne séquentiellement sur un seul processeur pour la matrice «helmo3» s'accroît avec l'augmentation de « ***N*** » . Quand ***N*** s'accroît de 25^3 à 80^3, la convergence devient de plus en plus difficile.

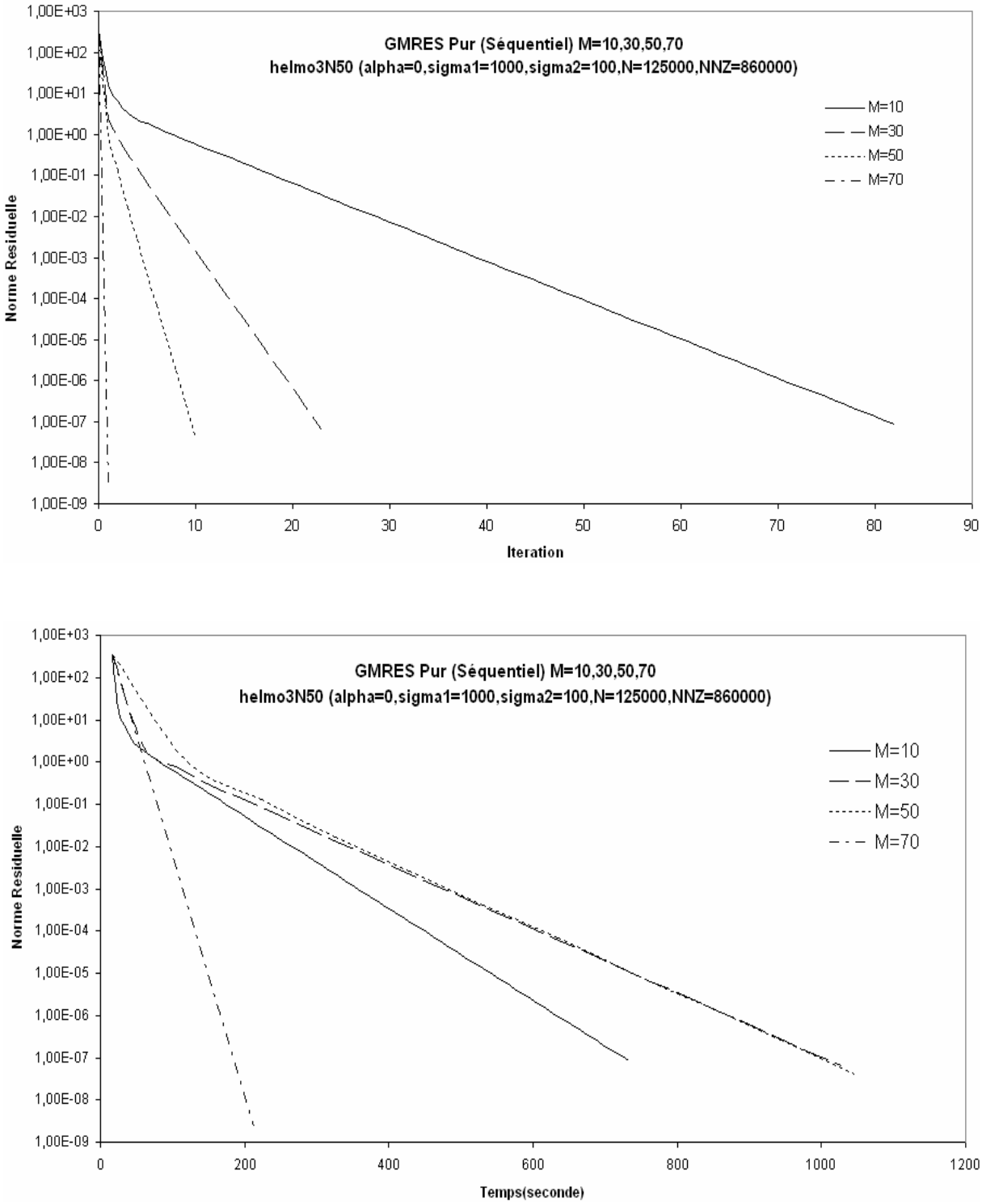


FIG 5.19-L'influence de la taille M du sous espace de Krylov de « helmo3N50 »

Dans cette figure FIG 5.19, la méthode traditionnelle GMRES(***m***) fonctionne séquentiellement sur un seul processeur pour la matrice «helmo3» avec le paramètre «$\boldsymbol{N} = 50^3$ » . Quand ***M*** s'accroît de 10 à 50, la convergence est de plus en plus lentement

atteinte. Au contraire, quand ***M*** atteint 70, la matrice converge le plus rapidement avec le moindre nombre d'itérations. La valeur optimale de ***M*** dans ce cas est 70.

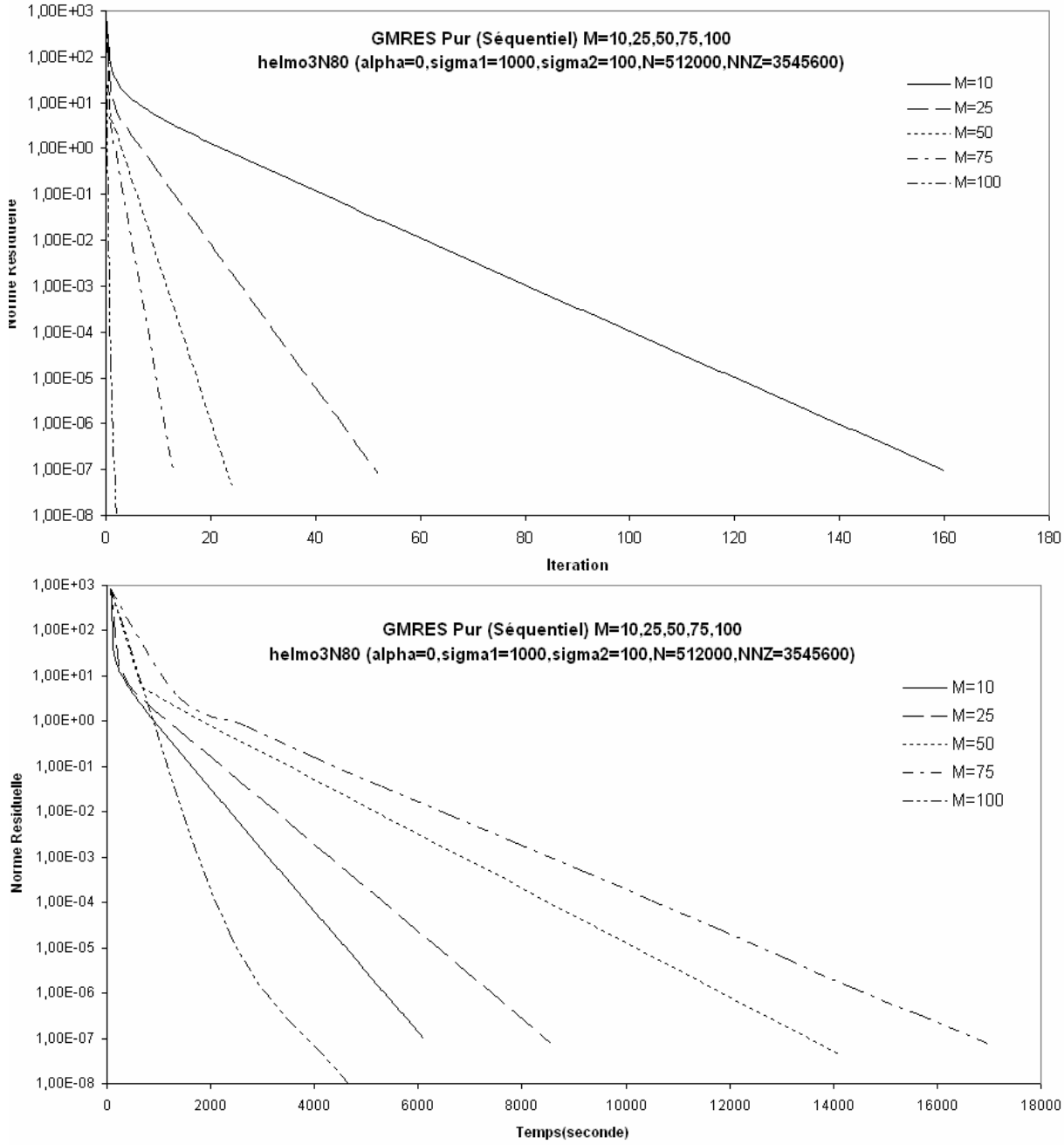


FIG 5.20-L'influence de la taille M du sous espace de Krylov de « helmo3N80 »

Dans cette figure FIG 5.20, La méthode traditionnelle GMRES(***m***) fonctionne séquentiellement sur un seul processeur pour la matrice «helmo3» avec le paramètre «$\boldsymbol{N} = 80^3$ » . Quand ***M*** s'accroît de 10 à 75, la convergence est de plus en plus lentement atteinte. Au contraire, quand ***M*** atteint 100, la matrice converge le plus rapidement avec

le moindre nombre d'itérations. Alors la valeur optimale de ***M*** dans ce cas est donc de 100.

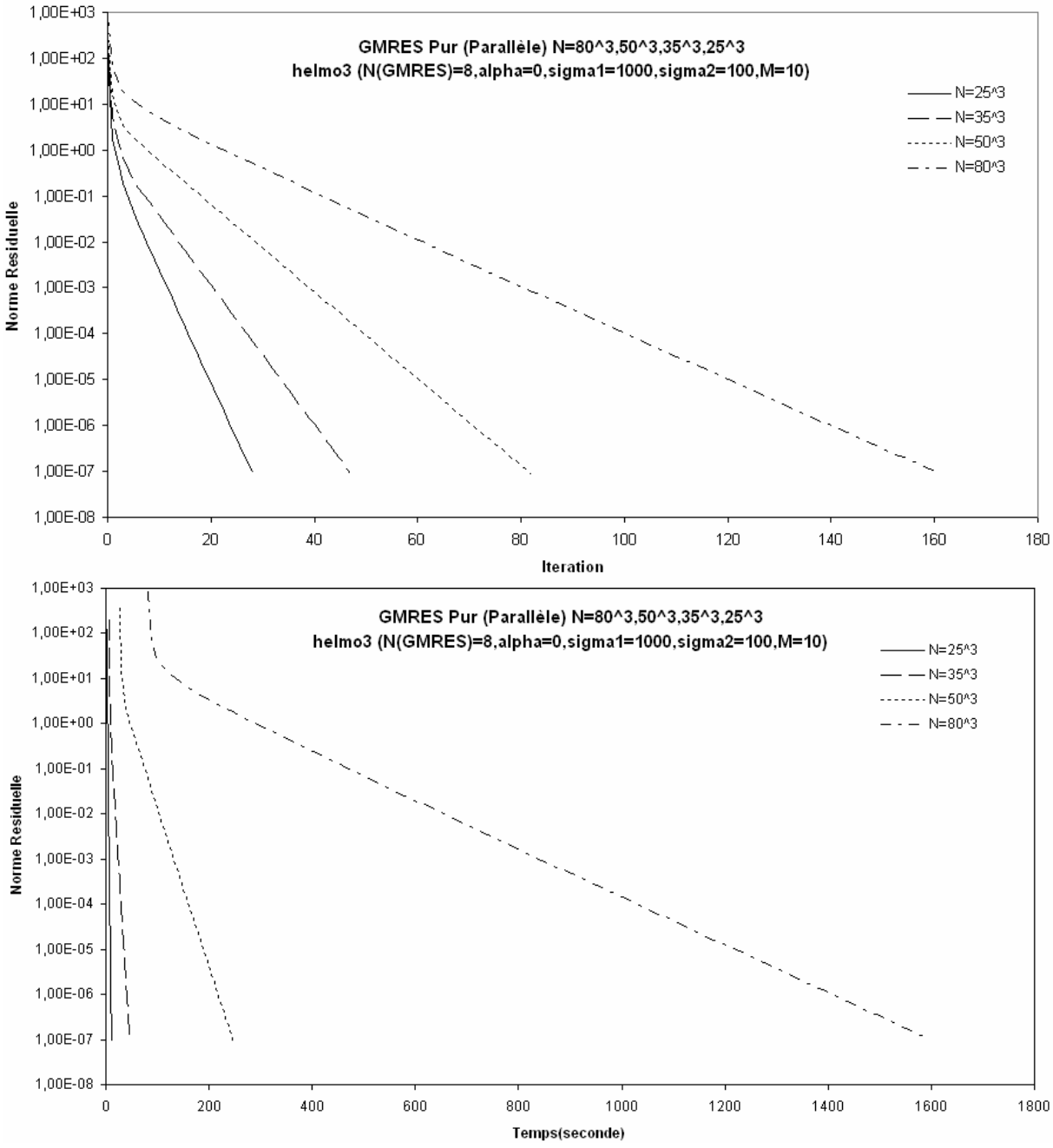


FIG 5.21-L'influence de la taille N des matrices « helmo3 »

Dans cette figure FIG 5.21, le temps permettant d'atteindre la convergence de la méthode traditionnelle GMRES(***m***) qui fonctionne parallèlement sur 8 processeurs pour la matrice «helmo3» s'accroît avec l'augmentation de « ***N*** » . Quand ***N*** s'accroît de 25^3 à 80^3, la convergence devient de plus en plus difficile.

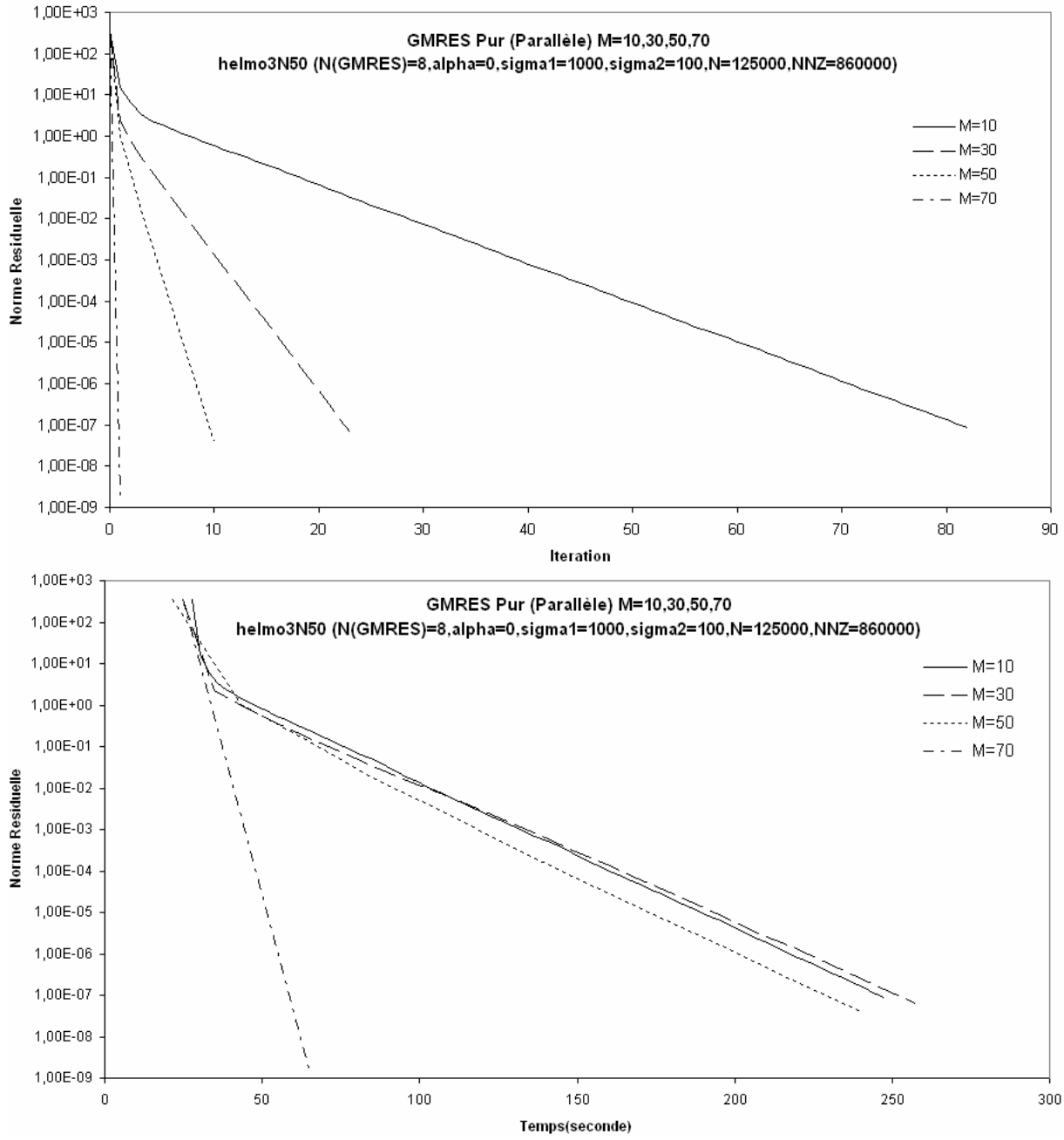


FIG 5.22-L'influence de la taille M du sous espace de Krylov de « helmo3N50 »

Dans cette figure FIG 5.22, La méthode traditionnelle GMRES(***m***) fonctionne parallèlement sur 8 processeurs pour la matrice «helmo3» avec le paramètre «$\boldsymbol{N} = 50^3$ ». Quand ***M*** s'accroît de 10 à 30, la convergence est de plus en plus lentement atteinte. Au contraire, quand s'accroît de 30 à 70, la matrice converge de plus en plus rapidement. Quand ***M*** vaut 70, la matrice converge le plus rapidement avec le moindre nombre d'itérations. La valeur optimale de ***M*** dans ce cas est donc 70.

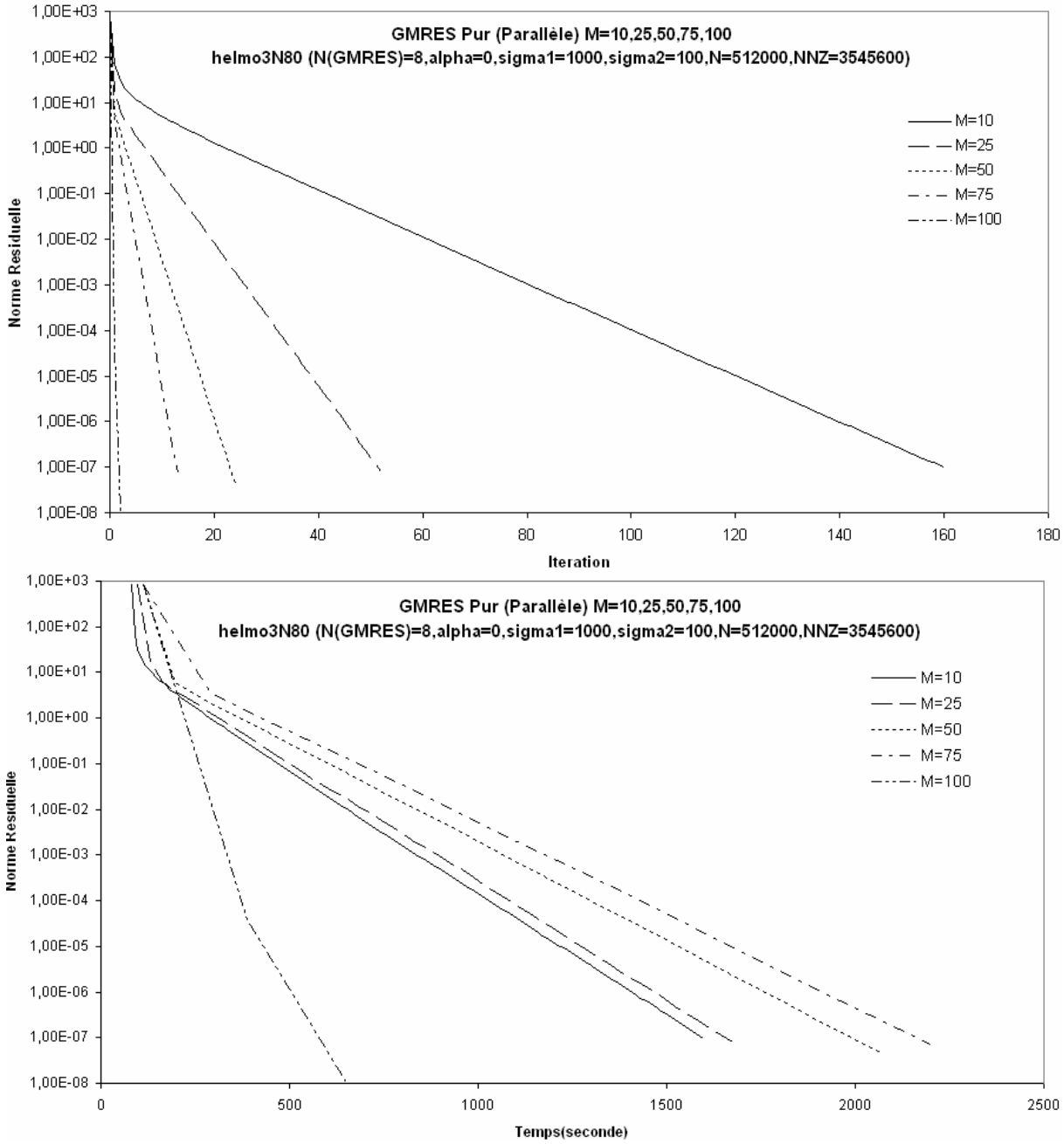


FIG 5.23-L'influence de la taille M du sous espace de Krylov de « helmo3N80 »

Dans cette figure FIG 5.23, La méthode traditionnelle GMRES(***m***) fonctionne parallèlement sur 8 processeurs pour la matrice «helmo3» avec le paramètre «$\boldsymbol{N}=80^3$ » . Quand ***M*** s'accroît de 10 à 75, la convergence est de plus en plus lentement atteinte. Au contraire, quand ***M*** atteint 100, la matrice converge le plus rapidement avec le moindre nombre d'itérations. La valeur optimale de ***M*** dans ce cas est donc de 100.

5.8 Conclusion

De nombreux problèmes scientifiques utilisant des données de grandes tailles ont souvent à résoudre un système linéaire : c'est fréquemment la partie la plus coûteuse en temps de calcul. C'est la raison pour laquelle il faut trouver le meilleur algorithme et améliorer les performances du calcul. En présence de matrices creuses de très grandes tailles, on utilise souvent des méthodes itératives parallèles telles que GMRES(*m*). L'accélération de ces méthodes conduit à proposer des méthodes hybrides telles que celle étudiée dans le cadre de cette thèse : la méthode hybride GMRES(*m*)/LS-Arnoldi, dans laquelle les valeurs propres obtenues par la méthode d'Arnoldi sont utilisées par la méthode « Least Squares » pour obtenir un itéré permettant un meilleur redémarrage de la méthode GMRES(*m*).

Nous avons donc adapté un logiciel développé initialement pour plusieurs machines communiquant entre-elles, écrit en Fortran et PVM, en un code parallèle asynchrone pour la machine IBM SP3, écrit en Fortran et MPI. Les expérimentations réalisées sur des matrices creuses à variables réelles de taille moyenne permettent de conclure à la possibilité d'accélérer la convergence en utilisant cette méthode. Nous avons aussi évalué expérimentalement l'importance de certains paramètres vis-à-vis de la convergence de la méthode. Nous n'avons pas utilisé de grandes matrices dès le départ car cela aurait posé des problèmes de temps de calculs ne permettant pas une série de tests importants. De plus, les matrices utilisées sont disponibles librement sur Internet et les résultats peuvent donc être comparés à ceux de nombreux autres logiciels. Dans le même esprit, nous n'avons présenté ici que des performances sur un seul nœud de SP3 bien que le code soit complètement construit pour fonctionner quel que soit le nombre de processeurs et de nœuds.

Lorsque la matrice est complexe non-Hermitienne, une optimisation permettant d'obtenir le produit scalaire polynomial, utilisé dans le cas Hermitien, n'est plus possible. Ceci car les valeurs propres ne sont alors plus conjuguées. Nous avons néanmoins essayé de trouver des heuristiques pour se ramener au cas précédent. Celles-ci, dans leurs versions actuelles, n'ont pas donné de résultats satisfaisants. Néanmoins, il existe encore plusieurs pistes permettant d'utiliser, pour résoudre un système linéaire, les valeurs propres calculées avec le logiciel parallèle développé, complètement asynchrone et exécutable

aussi bien sur cluster de SMP que sur NOW. Il est également envisageable d'hybrider ces calculs avec d'autres méthodes que GMRES, ainsi que de cibler d'autres préconditionnements polynomiaux ou diagonaux.

Nous avons alors concentré notre effort sur la méthode GMRES(*m*) elle-même, pour les applications du CEA dont nous avions un générateur des matrices associées. Le logiciel parallèle développé nous a donné la possibilité de tester la méthode GMRES avec des tailles de sous-espaces très grandes pour les matrices du CEA. Néanmoins nous avons mis en évidence que lorsque cette taille augmente, le temps de calcul « dans le sous-espace » peut devenir très important. En fait, pour une matrice et une machine donnée, il existe des valeurs « optimales » de ce sous-espace. Pour le premier problème traité, dont nous avons pu disposer du code du CEA générant la matrice associée, nous avons pu résoudre les problèmes ciblés avec GMRES seul, pour des valeurs de paramètres aboutissant à un échec dans les autres méthodes testées [32]. Nous avons bien mis en évidence l'influence de la taille de sous-espace à la fois sur la convergence et le temps de calcul d'une itération. Pour le second problème dont nous avions le code générant des matrices associées, nous avons obtenu des convergences très rapides avec GMRES simple, mais pour des tailles de matrices plus modestes que celles ciblées par le CEA, car le code de génération de ces matrices n'est pas parallèle et nous obtenions des temps de calcul prohibitifs pour générer les matrices en séquentiel. Le nombre de processeurs utilisés dans cette étude du cas complexe est modeste et sur un seul nœud de IBM SP3. Nous avons choisi de rester dans un cas de figure courant. Néanmoins, le logiciel peut, en l'état, tourner sur plusieurs nœuds de cluster de SMP pour réduire le temps de calcul ; mais pour cela il serait préférable de disposer d'un générateur de matrices parallèle.

Alors qu'il n'y a pas, a priori, d'autres possibilités d'améliorer les autres méthodes choisies, il est encore possible d'améliorer la qualité de la convergence de GMRES en faisant une réorthogonalisation systématique des vecteurs de la base de Krylov calculée dans GMRES ; sans parler de préconditionnement. De même, il est possible d'envisager de faire varier la taille du sous-espace au cours des redémarrages de la méthode, selon la variation du résidu et en fonction de la vitesse de convergence. Il est aussi possible de se servir de l'asynchronisme du logiciel pour exécuter plusieurs GMRES en parallèle avec des directions de projection différentes, tant que nous n'avons pas obtenu une décroissance suffisante du résidu. Il est aussi possible de continuer à augmenter la taille

du sous-espace en effectuant une orthogonalisation très partielle : i.e. on n'a pas besoin de conserver tous les vecteurs de la base et donc on n'obtient pas une augmentation importante de l'espace mémoire nécessaire pour stocker la base de l'espace de Krylov. Ceci pouvant être associé à une réorthogonalisation systématique entre les vecteurs conservés.

Si la méthode étudiée permet donc de bonnes accélérations lorsque la matrice est à coefficients réels, celle-ci ne fonctionne pas lorsque la matrice est à coefficients complexes non-Hermitiens. Dans ce dernier cas, il est néanmoins possible d'espérer résoudre les applications liées aux matrices ciblées.

D'un coté, il est nécessaire de pousser la recherche de possibles solutions mathématiques ou d'éventuelles heuristiques performantes permettant de faire fonctionner l'accélération présentée dans ce document dans le cas où la matrice est à coefficients à variables complexes. Par exemple, une étude de la répartition du spectre autour de l'axe réel, en partant des valeurs de Ritz obtenues à l'aide de la méthode d'Arnoldi, peut ensuite permettre de sélectionner un sous-ensemble de ces valeurs pour lesquelles l'heuristique proposée peut être intéressante. Néanmoins, cela demanderait probablement de calculer un plus grand nombre de valeurs de Ritz et nous n'avons pas encore approfondi la signification de cette heuristique au niveau mathématique, préférant ne pas nous écarter dans un premier temps de notre démarche très pragmatique. D'autant qu'il est souhaitable de prendre en compte les particularités des matrices ciblées par le CEA dans cette étude. Plutôt que de chercher une heuristique générale, il est préférable de se restreindre dans un premier temps, au regard de la difficulté du problème, à une sous-classe de matrices.

D'un autre coté, en remarquant que GMRES sans préconditionnement possède un potentiel encore important pour peu que la taille du sous-espace soit bien choisie et assez grande, il est envisageable de se contenter de variantes de cette méthode de base dans un premier temps. L'utilisation d'un nombre de processeurs important devrait réduire les temps de calcul et permettre de traiter de très grands problèmes, si nous possédons par ailleurs un générateur de matrice lui-même parallélisé.

Chapitre 6 Méthodes de Krylov sur GRILLE à Grande Echelle

6.1 Introduction

De nos jours, beaucoup de dispositifs avec ou sans fils tel que les Clusters, les PCs, même les PDAs, les portables, reliés ensemble par l'énorme réseau Internet, ont pu être largement dispersés géographiquement. Le bénéfice principal du calcul global est d'utiliser leur temps libre pour faire tourner une application très grande et répartie. Et la puissance de calcul est fournie par les dispositifs et les instruments volontaires qui accordent une certaine partie de leurs heures inexploitées à l'exécution d'une partie d'application avec les données associées. En ce mode, les ressources informatiques sont en fait une réunion hétérogène de GRILLE. Dans beaucoup d'aspects techniques chacun des dispositifs ou des instruments pourrait être considérablement différents des autres: Vitesse d'unité centrale de traitement, charge d'unité centrale de traitement, espace mémoire, configuration matérielle, système d'exploitation et débit de réseau, charge du réseau, etc. Quelques implantations ont déjà faites, qui utilisent Globus [33], le projet de SETI@home [36], XtremWeb [33],[35], [37]etc.

Le projet de XtremWeb vise à établir une plateforme pour étudier des modèles d'exécution dans le cadre général du calcul global. De même que le projet de calcul global SETI@home [36] ou Folding@home, le but de XtremWeb est de distribuer un groupe d'applications à un ensemble de dispositifs dans l'Internet ou l'Intranet par la méthode du vol de cycles, en particulier en se concentrant sur les applications de multiparamètres qui peuvent être exécutées de nombreuses fois avec des paramètres d'entrée différents. Chaque composant de calcul est totalement autonome [33].

Au sujet de MPI (Message Passing Interface) [38][39], il s'agit d'un ensemble de fonctions d'API permettant à des programmeurs d'écrire des programmes parallèles efficaces qui utilisent des échanges de messages entre une série de processus pour accomplir un travail parallèle global. MPI est le résultat des décennies de recherche sur le calcul parallèle, et a été créé par le forum de MPI - un groupe ouvert représentant une section transversale large d'intérêts industriels et scolaires. Davantage d'information, y

compris les deux volumes de la norme des fonctions MPI, peut être trouvée sur le principal site [40].

MPI convient aux machines parallèles de type "big iron" telles que SGI Origin, IBM SP, etc., mais fonctionne également dans de plus petits environnements tels qu'un groupe de postes de travail. Puisque les clusters des postes de travail sont aisément disponibles dans beaucoup d'établissements, il est devenu courant de les employer comme ressource informatique parallèle simplement en exécutant des programmes MPI. La norme de MPI a été conçue pour soutenir l'indépendance de portabilité et de plateforme. En conséquence, les utilisateurs peuvent apprécier des possibilités de développement trans-plateforme qu'une communication hétérogène transparente. Par exemple, des codes de MPI qui ont été écrits sur l'architecture RS-6000 ayant le système AIX peuvent être mis en communication avec une architecture de SPARC utilisant le système Solaris avec peu ou pas de modifications.

LAM-MPI est une implantation de haute performance, librement disponible, open source de la norme de MPI qui est étudiée, développée, et maintenue au laboratoire de systèmes ouverts de l'université de l'Indiana. LAM-MPI est compatible avec tous les standards MPI-1 et respecte une grande partie de la norme MPI-2. Plus d'informations sur LAM-MPI, y compris tous les codes sources et la documentation, sont fournies par le principal site[41].

LAM-MPI est non seulement une bibliothèque qui met en application les APIs de MPI, mais également l'environnement d'exécution de LAM : environnement d'exécution au niveau de l'utilisateur et basé sur un démon « runtime », il fournit plusieurs des services qui sont exigés par des programmes MPI. Les deux composants principaux du paquetage de LAM-MPI sont conçus en tant que cadres composants - extensibles avec de petits modules qui sont sélectionnables (et configurables) au moment de l'exécution. Ce cadre de composants est reconnu comme les System Services Interface (SSI). Les architectures des composants de SSI sont entièrement documentées dedans[42]-[45]. Nous avons implanté l'algorithme GMRES(*m*) sur l'environnement LAM-MPI 6.5.4 de Polytech-Lille.

Dans ce chapitre, nous présentons une version distribuée de l'algorithme GMRES (*m*) appliqué au système de GRID léger XtremWeb : un réseau local avec 128 PCs non spécialisés à Polytech-Lille (à l'université de Lille I en France), un réseau à distance avec une grille de 3 grappes de SCGN qui inclut 91 CPUs totalement au HPCC (High Performance Centre for Computational Science) de l'université de Tsukuba au Japon. Nous comparerons en outre les performances obtenues avec XtremWeb à ceux de LAM-MPI sur la même plateforme de LAN et à celles de l'IBM SP4 présenté dans la section 2.3 dans un contexte de supercomputing. Nous présentons les avantages et les inconvénients de nos réalisations sur ces trois systèmes de calcul.

6.2 Système de GRID XtremWeb

Le système léger de GRID XtremWeb [27][28][33],[35][37] qui vise à distribuer des applications aux ressources dynamiques et volatiles correspondant à leur disponibilité en appliquant ses propres politiques de tolérance aux fautes et de sécurité, est un logiciel plateforme de source ouvert, logiciel libre(GPL) et sans but lucratif pour les applications scientifiques de calcul global et le système distribué pair-à-pair [46].

Dans l'architecture de XtremWeb, pour participer au calcul global, les dispositifs peuvent jouer deux rôles :

Un Collaborateur : Un collaborateur est une machine puissante qui s'enregistre au serveur d'administration de XtremWeb. Elle peut télécharger le logiciel système entier de XtremWeb, et installer ensuite son propre environnement de calcul global distribué. Un accord est donné au serveur d'administration de XtremWeb par le collaborateur. Cet accord permet au serveur principal de XtremWeb d'exploiter les ressources supplémentaires qui sont rassemblées par le collaborateur. Dans ce modèle le groupe de dispositifs gouvernés par un collaborateur fera les travaux de calcul distribués par le serveur principal de XtremWeb seulement quand ils sont libres de tous travaux assignés par leur propre collaborateur.

Un Volontaire : Un volontaire est un dispositif ou un instrument qui s'enregistre au serveur d'administration de XtremWeb volontairement. Il télécharge les composants de calcul et les installe. Tant qu'il aura du temps libre, il contribuera à ces travaux de calcul. Une vue d'ensemble du cadre de calcul global de XtremWeb est donnée dans la FIG 6.1.

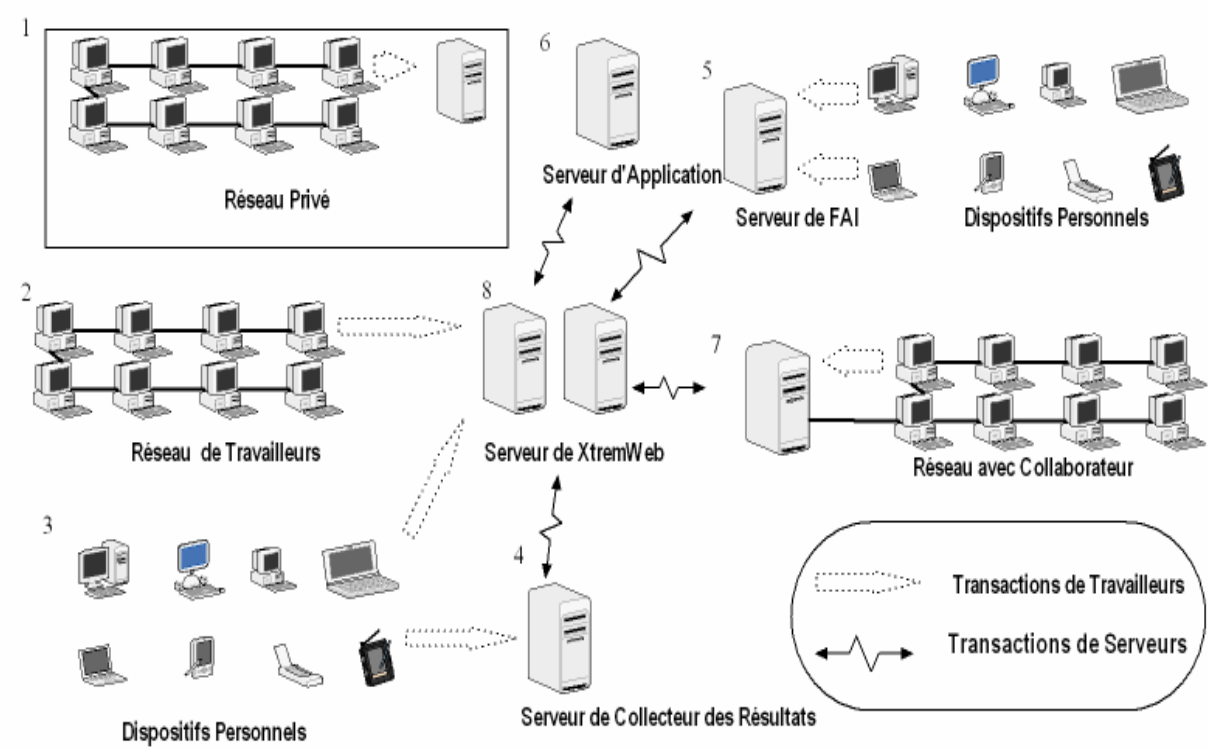


FIG 6.1Une vue d'ensemble de XtremWeb

Le serveur de XtremWeb est au centre de ce cadre (8). Il assigne les travaux de calcul et contrôle les communications. Un réseau local (2) de dizaines ou même de centaines de PCs dans une école ou une institution s'est relié au serveur de XtremWeb et son temps libre contribue au calcul global. Les collaborateurs peuvent administrer leurs propres applications réparties (6,7) et travaillent ensemble avec le serveur de XtremWeb tant qu'ils disposent de ressources disponibles. Un certain serveur peut être reconfiguré dans un but particulier tel que rassembler des résultats (4). Un modèle plus universel (5) peut effectuer le calcul global par l'intermédiaire des serveurs de FAI (Fournisseur d'Accès à l'Internet) qui se relient aux millions de dispositifs personnels (PC, Ordinateur Portable, PDA, iMac, Palmtop, etc). C'est un bon modèle commercial pour le calcul de GRID, parce que la durée de calcul consommée pour chaque dispositif peut être estimée et donc payée par le FAI. Nous pouvons aussi bien relier directement ces dispositifs personnels divers au serveur de XtremWeb pour faire notre calcul (3). Quoi qu'il en soit, une école, une entreprise ou un établissement peut installer un système de XtremWeb entièrement isolé au sein d'Intranet pour leurs propres demandes de calculs privés (1).

La gestion des travaux de XtremWeb est le modèle du Coordonnateur-Travailleur (voir FIG 6.2). Le coordonnateur s'occupe du processus de gestion des travaux. En particulier, il stocke tous les résultats, mais on peut regretter qu'aucun vrai concept de programmation de tâches n'est appliqué jusqu'ici. La seule politique de programmation mise en œuvre est de type FIFO (premier entré premier sorti). Les travailleurs sont les dispositifs volontaires distribués qui donnent leur temps de CPU libre pour effectuer les tâches de calcul fournies par le coordonnateur. C'est un modèle de "Pull": Tous les actions et connexions sont seulement initiées par les Travailleurs. Le Coordonnateur enregistre chaque connexion de Travailleur. Selon sa politique locale, le Travailleur demande des tâches au Coordonnateur pour faire des calculs. Et toutes les communications entre le Coordonnateur et les Travailleurs sont bien entendu chiffrées pour la sécurité de réseau.

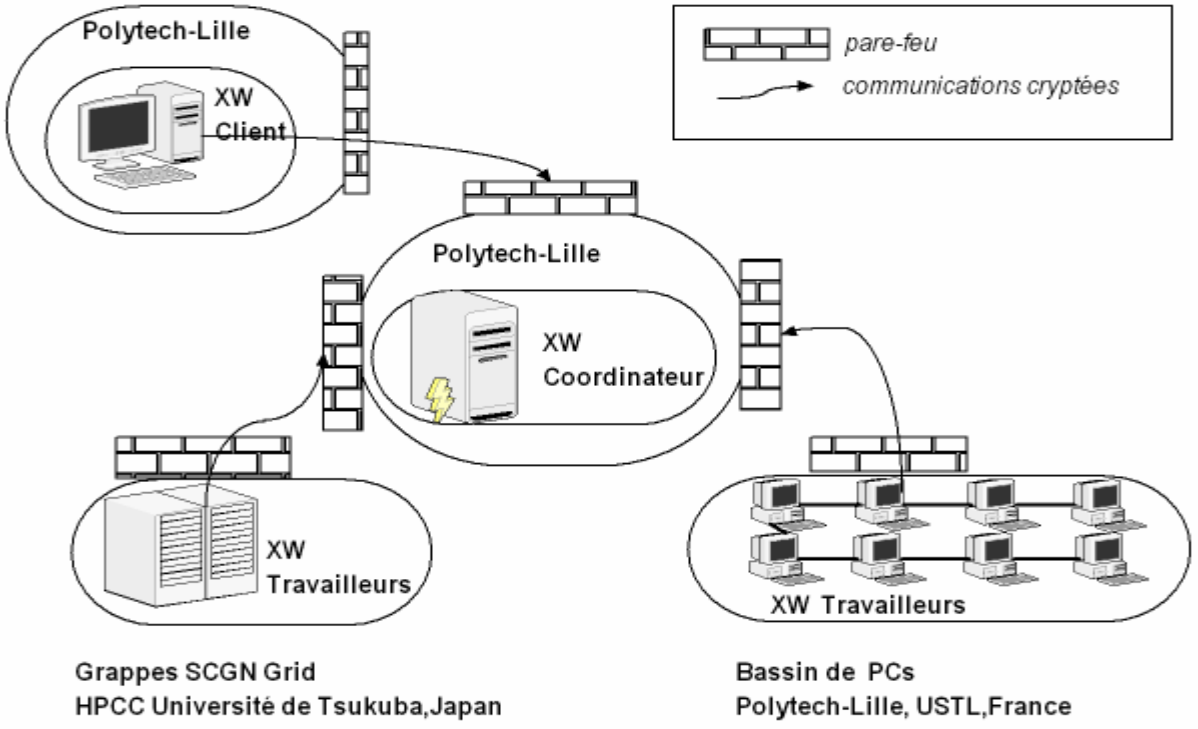


FIG 6.2 Réseau de XtremWeb pour l'exécution de GMRES

Le Travailleur télécharge le logiciel exécutable et tous les composants correspondants (les fichiers d'entrée, les arguments de ligne de commande pour le fichier binaire exécutable, etc..), les stocke sur des médias de stockage locaux, puis commence le calcul. Une fois qu'une tâche est finie, le Travailleur envoie les résultats (les fichiers de sortie) au Coordonnateur.

Pour la sécurité de réseau, pour garantir l'authentification de l'utilisateur, l'intégrité des Travailleurs, la protection de l'application et des résultats hébergés, XtremWeb dépend de trois mécanismes : la liste des utilisateurs autorisés, les authentifications du Coordonnateur par Travailleurs et par Clients, l'utilisation de « sandbox ». Il y a également un pare-feu sur chaque site pour rassurer la sécurité du réseau. Les protocoles de XtremWeb emploient (avec le modèle de « Pull » du côté du Travailleur) des communications unilatérales pour résoudre le problème du pare-feu.

6.3 Implantation de GMRES(m) sur XtremWeb

Dans notre implantation de l'algorithme GMRES (m) (qui est déjà bien présenté dans la section 3.2.3) sur XtremWeb, dans le but de gagner la mémoire et de réduire les communications sur le réseau, toutes les matrices creuses utilisées sont stockées dans le format compressé **CSR**. Ce format est le plus économe pour stocker les matrices creuses en mémoire (voir la section 3.1.2). Pour mettre en place cet algorithme, les tâches de calcul sont isolées et forment 11 composants logiciels. Dans la FIG 6.3 les composants dont les noms commencent par "Comm" sont exécutés sur le Coordonnateur de XtremWeb, ces programmes étant exécutés séquentiellement. Les composants dont les noms commencent par "Sub" sont exécutés sur des Travailleurs de XtremWeb d'une manière parallèle.

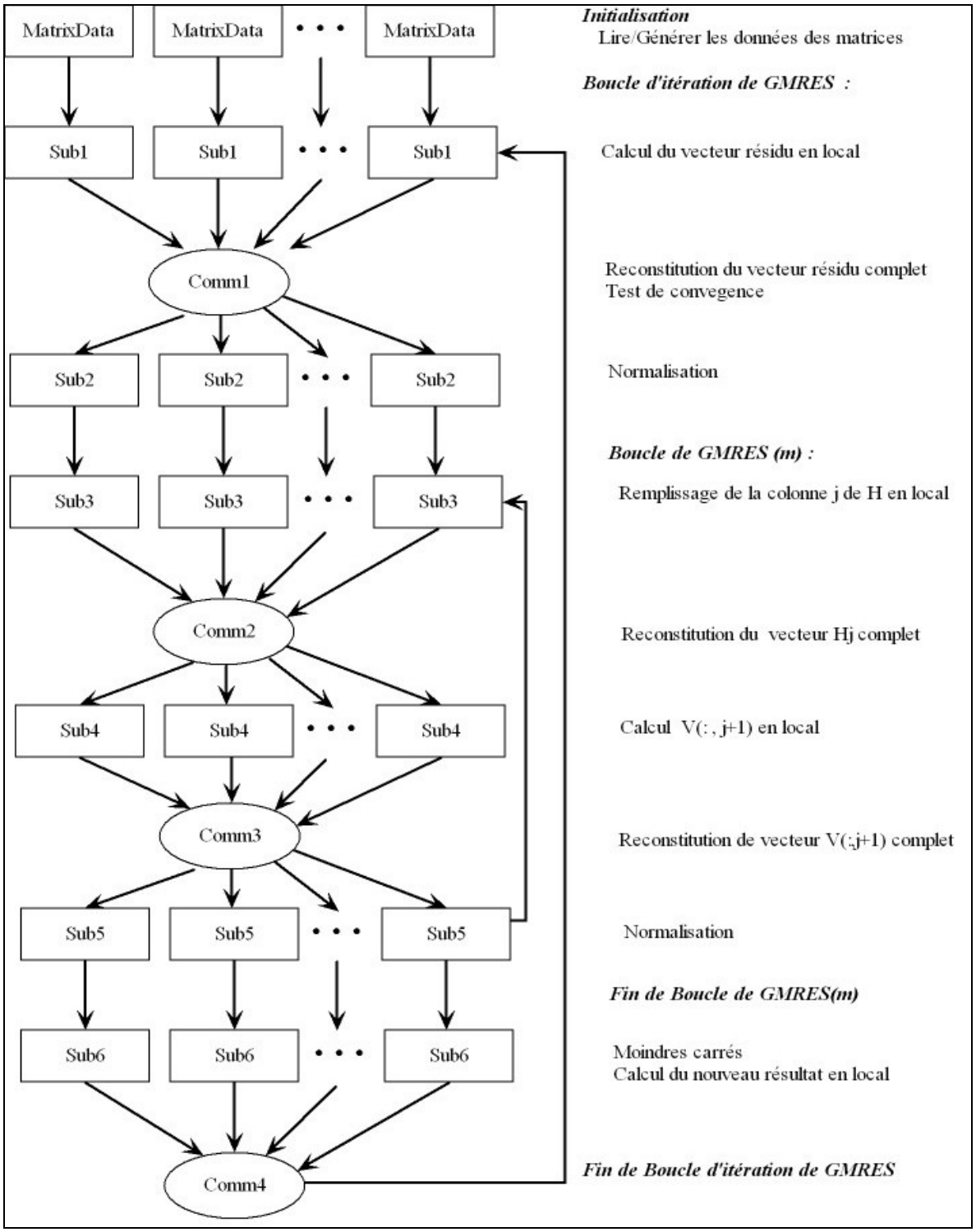


FIG 6.3 Graphe d'implantation de GMRES(m) sur XtremWeb

6.4 Résultats Numériques et Analyses

Deux configurations de réseau de XtremWeb sont employées pour nos essais: une de LAN et une de WAN. Le Coordonnateur des deux réseaux de XtremWeb tourne sur un serveur spécialisé dans le bâtiment de l'école d'ingénieur universitaire de Lille (Polytech-Lille) en France. La configuration locale, les Travailleurs (128 PCs, non spécialisés,

système d'exploitation Linux) tournent dans les salles d'ordinateurs de Polytech-Lille. La configuration de WAN emploie les 3 grappes de la grille de SCGN ♦ au HPCC (High Performance Centre for Computational Science) de l'Université de Tsukuba au Japon. En tout, 82 Travailleurs (CPUs dans les grappes) sont en service. La description des détails des deux sites est donnée dans la table TAB 6.1. Pour les mêmes matrices testées, on a fait également des essais sur une plateforme de supercalculateur IBM SP4, dont la configuration puissante est présentée dans la table TAB 6.2

TAB 6.1 Les réseaux de XtremWeb

Configuration locale à Lille (128 PCs)		
Nombre	CPU	Mémoire
28	Pentium III (Katmai), 450MHz	128MB
28	Intel Celeron, 2.2GHz	512MB
23	Intel Celeron, 2.4GHz	512MB
15	AMD Duron, 750MHz	256MB
14	Celeron (Coppermine), 600MHz	128MB
8	Intel Celeron, 2GHz	512MB
8	Intel Celeron, 1.4GHz	256MB
4	Pentium 4, 2.4GHz	512MB
Configuration de WAN à Tsukuba (3 Grappes)		
Nombre de Nœuds	CPUs de nœud	Mémoire par nœud
15	Dual P4 Xeon 3065MHz	1GB
10	Dual Athon 1533MHz	882MB
8	Quad P3 450MHz	2GB

TAB 6.2 La configuration de supercalculateur IBM SP4

IBM SP4 (288 CPUs)		
Nombre de Nœuds	CPUs de nœud	Mémoire par nœud
2	32 Power 4 1.3GHz	64GB
2	32 Power 4 1.7GHz	64GB
5	32 Power 4 1.7GHz	32GB

♦ http://www.omni.hpcc.jp/ganglia/

La première matrice examinée est la matrice "vp" avec la taille $2q \times 2q$ présentée dans la FIG 6.3, ses valeurs propres sont $a_j + ib_j$, et nous prenons les valeurs propres dans deux rectangles R_1 avec les sommets $-2.5 \pm 2i$, $-1.5 \pm 2i$, et R_2 avec les sommets $1.2 \pm 4i$, $1.8 \pm 4i$ [48]. Dans nos essais, nous employons une matrice "vp" produite avec la taille 100 000 archivée dans un fichier au format MatrixMarket que nous nommerons dans la suite "matrixvp1E05". Une matrice "vp" produite par un générateur avec la taille 10^6 , s'appellera dans les tables suivantes "matrixvp1E06".

$$A = \begin{Bmatrix} a_1 & \frac{b_1}{2} & & & & & & \\ 2b_1 & a_1 & & & & & & \\ & & a_2 & \frac{b_2}{2} & & & & \\ & & 2b_2 & a_2 & & & & \\ & & & & \bullet & \bullet & & \\ & & & & \bullet & \bullet & & \\ & & & & & & a_q & \frac{b_q}{2} \\ & & & & & & 2b_q & a_q \end{Bmatrix}$$

FIG 6.4 Matrice Creuse VP

La deuxième matrice est la matrice "DA" qui est une matrice dense et carrée produite par un générateur de matrice. "DA10000" est une matrice de dimension 10 000 comprenant 10^8 éléments non nul. "DA20000" est une matrice de dimension 20 000 comprenant 4×10^8 éléments non nul.

La troisième matrice examinée est une matrice téléchargée du site "MatrixMarket". "ADD200" est une matrice réelle carrée creuse non symétrique de dimension 2359, incluant 171319 éléments non nuls.

Le but de nos essais n'est pas de mesurer les performances sur chaque pair ni celles de notre système de GRID XtremWeb. En effet ce serait un ensemble de données littéralement sans valeur qui seraient probablement toujours périmées. En fait, en raison de la nature dynamique et volatile de la grille, ce genre d'information n'est pas exploitable et n'a d'intérêt que pour une programmation en temps réel. Néanmoins, grâce aux résultats d'essai que nous obtenons finalement, nous pourrions recommander XtremWeb - un bon système de GRILLE pour certaines résolutions de calcul numériques.

Dans le monde réel, la durée du calcul varie dans de grandes proportions (la fréquence du processeur diffère, de même que la mémoire disponible, la charge de l'unité centrale de traitement, le trafic du réseau, etc…, et aussi bien que la volatilité élevée des Travailleurs de XtremWeb).

*TAB 6.3 DA20000 avec 15 Travailleurs sur LAN (N=20000,NNZ=20000*20000 Lille, Générateur)*

Temps de calcul (CPU)	24,87s	21,96s	22,78s	6,93s
Temps de calcul (Coordinateur)	4390,64s	1958,84s	2624,44s	591,02s
Temps de communication	1029,46s	1028,54s	1027,02s	1024,62s
Temps Total	5420,10s	2987,38s	3651,46s	1615,64s

Dans la table TAB 6.3, quant à la durée de calcul (temps de CPU) que nous relevons localement sur le Travailleur, c'est le vrai temps de CPU consommé pour notre calcul pur sur chaque Travailleur. Pour les travaux parallèles, nous choisissons dans le groupe entier le temps du travail qui pénalise donc la performance entière du groupe. Quant au temps de calcul que nous évaluons sur le Coordonnateur, il est mesuré par la différence entre le moment où le Coordonnateur envoies un groupe de travaux parallèles et le moment où il reçoit le signal "fini" de tous les travaux parallèles dans ce groupe. En fait, ce temps inclut les attentes dues aux synchronisations entre les Travailleurs. Le temps de communication est la différence entre le temps de calcul (Coordonnateur) et le temps total. Nous employons les mêmes termes pour les tables suivantes.

Nous pouvons remarquer des variations importantes des performances concernant des calculs analogues : Puisque nous ne pouvons pas choisir exactement les Travailleurs que nos travaux utilisent, quand ceux-ci sont exécutés accidentellement sur les Travailleurs les plus puissants accidentellement, le temps de CPU utilisé est nettement plus réduit (6.93s), mais même lorsque les puissances des Travailleurs sont similaires (les temps d'unité centrale de traitement utilisés sont tous d'environs 22 secondes), la charge de réseau et son statut ont une influence importante sur les performances finales. C'est pourquoi toutes les données présentées dans nos tables sont des moyennes entre les résultats de plusieurs tests.

Un facteur important est la configuration du réseau (TAB 6.1) lors de nos essais. Nous avons fait quelques essais afin de comparer les performances différentes entre la

configuration de LAN et celle de WAN. Dans les tables TAB 6.4 et TAB 6.5, nous remarquons un temps total visiblement plus long dans la configuration WAN. Il peut s'expliquer par le fait que le débit du WAN est plus bas que celui du LAN, mais la différence n'est pas si considérable grâce à la connexion de haut débit entre Lille et Tsukuba. Nous remarquons également que moins de temps de calcul de l'unité centrale de traitement est consommé pour nos travaux concernant la configuration du WAN. Ce résultat peut être expliqué par les processeurs plus puissants et les mémoires plus grandes des grappes à Tsukuba. Dans la pratique, pour les programmes qui exigent une grande mémoire ou produisent beaucoup de communications, nous employons la configuration de WAN profitant de processeurs plus puissants, des mémoires vive plus grandes et de Travailleurs moins volatils.

*TAB 6.4 DA10000 de XTREMWEB de LAN (N=10000, NNZ=10000*10000, M(GMRES)=8,ITERATIONS=3, Lille ,Générateur)*

	nT=5	nT=10	nT=20	nT=25
Tâches	400	800	1600	2000
Temps de CPU	4,88s	2,51s	1,39s	1,10s
Temps Total	894,10s	863,71s	1394,23s	1696,73s

*TAB 6.5 DA10000 de XTREMWEB de WAN (N=10000, NNZ=10000*10000, M(GMRES)=8,ITERATIONS=3, Tsukuba ,Générateur)*

	nT=5	nT=10	nT=20	nT=25
Tâches	400	800	1600	2000
Temps de CPU	2,59s	1,31s	0,76s	0,61s
Temps Total	1793,47s	1059,51s	1581,55s	1760,57s

La politique utilisée pour stocker nos matrices lors de notre calcul est un autre facteur principal influant sur les performances de nos essais. Deux stratégies ont été employées. Une façon est d'utiliser les fichiers de données lus par tous les sous-programmes dans chaque étape de notre algorithme, solution qui s'impose concernant les matrices téléchargées d'un site Internet, notamment avec le format de MatrixMarket. Évidemment, cette méthode causera beaucoup de trafic sur le réseau en raison du volume important des données de la matrice transférées pour chaque composant de calcul distribué. L'autre manière est de construire de la matrice localement par un générateur de matrice en version distribuée là où nous avons besoin des données de la matrice pour les calculs. En fait, dans la FIG 6.3, seuls deux composants « Sub1 » et « Sub3 » exigent les

données de la matrice pour les calculs. Évidemment, le trafic du réseau est alors nettement moindre, particulièrement pour une matrice de grande taille possédant beaucoup d'éléments non nuls. Mais il est obligatoire dans ce cas de disposer un générateur de matrice distribué, et concernant certaines des matrices présentes nous n'avons pas pu trouver un tel générateur.

TAB 6.6 Matrice VP100000 avec 10 Travailleurs sur WAN (M(GMRES)=30,Itérations=5, 3730 Tâches,Tsukuba)

	Avec Fichier	Avec Générateur
Temps de calcul (CPU)	2,71s	2,89s
Temps de calcul (Coordinateur)	33223s	26252s
Temps de communication	61947s	55482s
Temps Total	95170s	81734s

Dans la TAB 6.6, concernant le temps de calcul (CPU), nous pouvons noter peu de temps de calcul (CPU) consommé grâce aux processeurs puissants des grappes et au volume modeste des données de cette matrice (pour chaque Travailleur, la matrice entière "matrixvp1E05" est coupée en 10 morceaux avec la dimension de 10 000 comprenant 20 000 éléments non nuls). Nous pouvons noter également que nous gagnons presque de 15% de performance dans ce cas-ci grâce à la diminution significative des communications et du trafic des réseaux. Pour une plus grande matrice, nous espérons une performance encore meilleure.

Les résultats de la table TAB 6.7 montrent que quand nous augmentons nT (le nombre de Travailleurs participant au calcul global), nous obtenons une granularité plus fine, ce qui veut dire que pour chaque tâche exécutée, on consomme moins de temps de calcul (CPU), alors que simultanément le nombre de tâches augmente linéairement en fonction du nombre de Travailleurs. Le temps de CPU consommé par le calcul diminue avec davantage de Travailleurs grâce au plus petit volume des données allouées à chaque tâche de calcul. Quant au temps total que nous évaluons sur le Coordonnateur, on observe une valeur optimale nT=25 la table TAB 6.7 quand nT=25, nous obtenons le temps de calcul le plus court du côté du Coordonnateur. Quand nT est petit, l'essentiel du temps est consacré au calcul (dans cette table « nT=10 », « nT=15 » sont dans ce cas). Mais le trafic de réseau augmente considérablement avec plus de Travailleurs impliqués dedans. Nous pouvons remarquer que le temps de communication augmente beaucoup

quand nous servons plus de Travailleurs pour nos tâches. En tout, le temps total permettant d'obtenir la convergence pour notre méthode GMRES (m) augmente avec l'accroissement du nombre des Travailleurs après un seuil optimal. Ce résultat correspond à la théorie. En effet, l'implantation parallèle de GMRES (m) exige des communications intensives et des synchronisations multiples.

*TAB 6.7 DA20000 de XtremWeb de LAN (N=20000, NNZ=20000*20000, M(GMRES)=8,Itérations=3,Lille,Générateur)*

	nT=10	nT=15	nT=20	nT=25	nT=30
Tâches	800	1200	1600	2000	2400
Mémoire	480,75Mb	320,61Mb	240,66Mb	192,64Mb	160,47Mb
Temps de calcul (CPU)	45,88s	7,38s	5,21s	4,71s	3,37s
Temps de calcul (Coordinateur)	3371,69s	2291,64s	1429,06s	718,76s	1697,73s
Temps de communication	706,58s	1027,78s	1423,89s	1893,30s	2444,57s
Temps Total	4078,27s	3319,42s	2852,95s	2612,06s	4142,30s

Nous avons fait une série d'essais pour comparer les performances des deux plateformes de calcul: Environnement d'IBM SP4 MPI, environnement de LAM-MPI (voir la section 6.1). Nous indiquons les résultats dans les tables TAB 6.8 et TAB 6.9, le temps de calcul est mesuré par la fonction de MPI "MPI_WTIME( )" comprenant la charge de la synchronisation requise pour obtenir le temps universel. Pour la matrice de grande taille "matrixvp1E06" de dimension un million, à cause de la mémoire limitée sur les postes de travail de LAN Polytech-lille, les essais n'ont pas été faits sur le système XtremWeb. Dans la table TAB 6.8, on constate que le temps de calcul diminue avec plus de processeurs participants en raison de moins de données par calcul. Évidemment, davantage de processeurs impliqués causeront des communications plus intensives qui entraîneront plus de temps de communication. Il y a un équilibre à trouver entre ces deux tendances opposées. Quand le nombre de processeurs est 10, nous découvrons l'optimal. Un point important doit être mentionné : quand nous faisons des essais qui durent un peu longtemps avec beaucoup de postes de travail, particulièrement dans le cas de "nP=30", parfois le système se bloque après le démarrage du programme. Nous avons donc fait un contrôle : nous avons alors constaté que quelques machines dans la liste des postes de travail engagés pour le calcul de LAM-MPI étaient « perdues ». Nous avons dû nous débarrasser de ces postes de travail qui tombaient en panne, puis nous avons rechargé le système LAM-MPI pour remettre en marche notre programme. Ce point

indique un désavantage de LAM-MPI : il n'y a aucune tolérance de fautes quand un poste de travail dans le réseau informatique ne fonctionne plus.

TAB 6.8 Matrice VP1E06 de LAM-MPI de LAN (N=1E06, NNZ=2E06, M(GMRES)=50,Itérations=38, Lille, Générateur)

	nP=4	nP=10	nP=20	nP=30
Temps de calcul	543,09s	235,43s	141,04s	108,00s
Temps de communication	2775,62s	2871,64s	4484,44s	7541,42s
Temps Total	3318,71s	3107,06s	4625,49s	7649,42s

TAB 6.9 Matrice VP1E06 de IBM SP4 MPI (N=1E06, NNZ=2E06, M(GMRES)=50,Itérations=38,Tsukuba, Générateur)

	nP=4	nP=10	nP=20	nP=30
Temps de calcul	727,33s	311,08s	159,86s	118,38s
Temps de communication	224,13s	220,82s	190,76s	186,24s
Temps Total	951,46s	531,90s	350,62s	304,62s

Dans la table TAB 6.8, comparé avec la table TAB 6.9, nous remarquons que, pour IBM SP4, le temps de calcul est peu plus important qu'avec LAM-MPI, ce qui peut être expliqué par les processeurs plus puissants des PCs à Polytech-lille. Dans la liste de machines de LAM-MPI, nous choisissons toujours pour nos essais les PCs avec les processeurs plus puissants et les mémoires les plus étendues. En fait, les PCs sur lesquels nous faisons nos essais ont des fréquences d'horloge au moins 2 Gigahertz et sont équipés d'une mémoires minimale de 512 MB. Tandis que l'IBM SP4 est équipé de CPUs qui sont moins rapides (maximum 1.7GHz dans la table TAB 6.2). Mais le temps de communication de l'IBM SP4 est beaucoup plus réduit que celui de LAM-MPI grâce à très haut (protocole US Fédération Switch au même noeud 1201Mb/s [49]) entre les processeurs de ce supercalculateur, Toutefois le débit maximum du LAN à Polytech-lille n'est que 10Mb/s. et grâce à la performance matérielle excellente du supercalculateur IBM SP4, même lorsque nous portons le nombre de processeurs pour les calculs parallèles de 4 à 30, le temps de communication diminue également. Par conséquent, nous avons besoin globalement de moins de temps pour atteindre la convergence de la méthode de GMRES(*m*).

Nous avons également réalisé quelques essais de "DA20000" avec l'IBM SP4 et la plateforme LAM-MPI : les résultats sont présentés dans les tables TAB 6.10 et TAB 6.11. Dans l'environnement de calcul de LAM-MPI et le système XtremWeb (voir les tables TAB 6.10 et TAB 6.7), quand le nombre de Travailleurs ou de processeurs utilisés est 10,

la mémoire requise pour chaque travail de calcul est environ 480Mb, mais pour la configuration LAN de Polytech-lille (voir TAB 6.1), la plus grande mémoire disponible est d'environ 512 Mb, et aucun poste de travail du réseau local n'est spécialisé, c'est-à-dire que toutes les grandes tâches, même les plus importantes, doivent être exécutées sur les PCs dont le CPU est administré en temps partagé. Nos tâches très gourmandes posent un problème. Une tâche exigeant beaucoup de mémoire cause également des paginations fréquentes, ce qui explique que dans ces deux tables, le temps de calcul concernant le cas de « nP=10 » est extrêmement plus long par rapport aux autres cas. Dans la table de LAM-MPI, le temps de calcul consommé est plus important que celui de XtremWeb parce que la fonction mesurant le temps universel "MPI_WTIME ( )" provoque un surcoût lié à la synchronisation, Il est aussi inférieur à celui de l'IBM SP4 grâce à des processeurs plus puissants comme indiqué dans l'analyse ci-dessus. Le temps de communication de LAM-MPI continue à augmenter avec davantage de processeurs sauf dans le cas spécial de "nP=10".

Au contraire, celui de l'IBM SP4 diminue sans interruption avec plus de processeurs engagés au calcul comme nous avons expliqué plus haut. Particulièrement, sur le système de l'IBM SP4 quand le nombre de processeurs utilisés est 10, nous n'avons pas un tel problème de mémoire qui ralentit le système entier comme avec LAM-MPI et XtremWeb grâce à la mémoire suffisante (dans la table TAB 6.2, on remarque que chaque processeur a au moins une mémoire de 1 Gigaoctet)

*TAB 6.10 DA20000 de LAM-MPI de LAN (N=20000, NNZ=20000*20000, M(GMRES)=8,Itérations=3,Lille,Générateur)*

	nP=10	nP=15	nP=20	nP=25	nP=30
Mémoire (Mb)	485,54	345,80	265,85	217,82	185,65
Temps de calcul (s)	450,65	7,55	5,39	4,31	3,59
Temps de communication(s)	558,13	20,78	33,33	54,10	77,96
Temps total(s)	1008,78	27,33	38,72	58,41	81,55

*TAB 6.11 DA20000 de IBM SP4 (N=20000, NNZ=20000*20000, M(GMRES)=8,Itérations=3,Générateur)*

	nP=10	nP=15	nP=20	nP=25	nP=30
Temps de calcul (s)	12,88	8,67	7,26	6,13	4,74
Temps de communication(s)	15,26	12,53	12,25	10,85	7,95
Temps total(s)	28,14	21,20	19,71	16,98	12,69

6.5 Conclusion

Nous avons mis en place notre algorithme GMRES(*m*) sur trois plateformes de calcul scientifique parallèle : Un système de grille XtremWeb, un environnement de calcul MPI populaire LAM-MPI et un système de supercalculateur IBM SP4. Ces plateformes sont évidemment très différentes.

Sans aucun doute, le système de supercalculateur IBM SP4 obtient les meilleures performances grâce à sa configuration excellente aussi bien matérielle que logicielle. Mais il souffre d'une imperfection principale : il est trop cher. Habituellement un lycée banal ou une petite société manque de moyens pour s'installer un tel système.

Le système de LAM-MPI montre une performance assez bonne avec un réseau local de postes de travail peu coûteux. Mais faisant face à la volatilité d'un réseau informatique global avec des milliers de dispositifs, son manque de tolérance aux fautes deviendra un maillon faible.

Le système de grille légère de calcul XtremWeb que nous présentons principalement en ce chapitre se révèle vraiment un bon système pour le calcul de grille. Comparé avec les supercalculateurs, le temps de résolution du problème est beaucoup plus important. Mais l'intérêt du modèle de calcul de grille est de profiter des postes de travail peu coûteux existant dans les salles informatiques ou à la maison, reliés ensemble par Internet, et le temps libre de ces ressources inexploitées. Confrontant le caractère volatil du réseau informatique global avec des milliers, voire beaucoup plus, de dispositifs reliés ensemble par Internet, XtremWeb a une bonne compétence de tolérance de fautes. Dans la configuration du réseau de XtremWeb, nous observons que XtremWeb est bien adapté à l'environnement hétérogène qui signifie des processeurs différents et des systèmes d'exploitation divers. Mais nous observons également que les composants séquentiels exécutés sur le Coordonnateur de notre logiciel de calcul de GMRES(*m*) (voir FIG 6.3), en sus de la gestion centrale de toutes les communications deviendront un embouteillage. La communication pourrait être optimisée et mise en place en mode décentralisé. Et une programmation plus efficace, au lieu du mode de FIFO pour les tâches distribuées aux Travailleurs, pourrait améliorer les performances de manière significative. Nous

estimons, en effet, que les travaux de calcul durant les plus longtemps obtiendront de meilleures performances avec la charge relativement plus légère de communication.

Les résultats de nos tests confirment d'ailleurs la portabilité de notre implantation sur le système XtremWeb. À l'avenir, avec plus de Travailleurs et avec un réseau à très haut débit, nous résoudrons des systèmes linéaires à très grande échelle de la taille de plusieurs millions sans difficulté.

Chapitre 7 Conclusion et Perspectives

Pour les nombreux problèmes scientifiques utilisant des structures de matrices de grande taille, la partie principale du temps de calcul est consommée par la résolution de systèmes linéaires. Nous devons donc souligner l'importance de l'amélioration de ce type de résolution.

La première partie de cette thèse présente la mise en œuvre des méthodes numériques itératives, pour des problèmes dans le cas réel décrits par des matrices creuses de grande taille dans des environnements parallèles surtout dans les environnements des instrumentations très puissantes et avancées : les supercalculateurs IBM SP3 et SP4. Nous montrons aussi quelques formats de compressions utilisés pour économiser les espaces de mémoire. Après avoir présenté l'intérêt des méthodes itératives par rapport aux méthodes directes, particulièrement quand il s'agit de matrices creuses, nous choisissons la méthode GMRES(*m*) comme la base de notre résolution.

Après avoir réalisé une version parallèle de la méthode GMRES(*m*) pure qui nous servira de référence, nous remarquons les limitations de son degré de parallélisme : quand nous augmentons la granularité de cette méthode avec davantage de processeurs participants, les performances se détériorèrent après avoir dépassé un certain seuil. Il existe donc du parallélisme disponible dont nous allons profiter. Nous avons ainsi développé ensuite une méthode hybride GMRES(*m*)/LS-Arnoldi. Dans cette méthode hybride, nous utilisons les valeurs propres récupérées par la méthode d'Arnoldi pour calculer les paramètres de la méthode « Least Squares » qui nous permettra d'obtenir un itéré donnant un meilleur vecteur de redémarrage de la méthode GMRES(*m*). Finalement nous obtenons ainsi une accélération de la convergence. Les aspects numériques de la théorie pour notre méthode sont détaillés dans la thèse de Azeddine Essai[31].

L'implantation parallèle de cette méthode hybride a été réalisé dans la thèse de Guy BERGERE [48] en utilisant principalement la bibliothèque parallèle PVM [52]sur deux sites offrant les accès à des machines parallèles différentes. Une version, utilisant deux machines parallèles (une ferme d'Alphas et une MasPar MP1[53],[54]) et deux stations séquentielles, forme le modèle « parallèle hétérogène asynchrone ». L'autre, utilisant une

machine parallèle (une CM5) [9] et trois stations de travail séquentielles a été qualifiée de modèle « parallèle entrelacée ».

Notre dernière implantation parallèle de cette méthode hybride dans cette thèse utilise les deux supercalculateurs IBM SP3 et IBM SP4 [7] avec la bibliothèque parallèle MPI [39]qui fonctionne à une couche du réseau plus haute que PVM. L'algorithme de la méthode hybride réalisé en MPI dans le contexte de supercalculateur est plus efficace et rapide.

Nous mettons en œuvre cette méthode hybride dans un environnement de calcul parallèle dans deux cas. Dans celui des matrices réelles, la théorie a été développée dans [31]. Dans le cas des matrices complexes, en l'absence du support d'une théorie mathématique, nous avons tenté d'utiliser la même méthode hybride et de trouver de manière empirique, en faisant de nombreux tests dans le Chapitre 5, les paramètres convenant à une résolution plus efficace.

La seconde partie de cette thèse est consacrée à la méthode GMRES(*m*) et à la méthode hybride GMRES(*m*)/LS-Arnoldi dans le cas complexe. D'abord, nous avons adapté les formats des matrices réelles pour les matrices complexes, puis nous avons modifié l'algorithme afin de s'adapter au cas complexe. Enfin, nous avons fait de nombreux tests avec des matrices téléchargées depuis le site MatrixMarket ainsi qu'avec les grandes matrices « SCH » et « HELMO » fournies par le CEA (*Commissariat à l'Énergie Atomique*). En dépit de résultats décevants, nous pensons que ces essais devaient être tentés en attendant de développer une théorie mathématique difficile à propos des matrices creuses complexes qui sont largement utilisées dans les applications des industries électroniques et électriques.

La troisième partie de la thèse étend la méthode GMRES(*m*) traditionnelle dans une autre direction : Les grilles de calcul, le calcul global et les systèmes pair-à-pair proposent l'intégration des ressources informatiques à large échelle. Les applications pair-à-pair et les expériences de calcul global ont montré la puissance potentielle énorme et les disponibilités du partage de ressources presque inépuisables sur Internet.

Comme les systèmes de calcul global (SETI@Home, distributed.net) et les applications pair-à-pair (napsters, gnutella), le système de grille léger XtremWeb propose l'utilisation massive des ressources inexploitées des réseaux locaux et d'Internet pour les applications distribuées et parallèles. XtremWeb est un système généraliste, il n'est pas dédié à une application spéciale, ni à une classe d'applications particulière. La destination d'XtremWeb est d'établir une plate-forme généralisée permettant de faire du calcul global avec production de calcul haute performance.

LAM-MPI est une implantation de haute performance, librement disponible, open source de la norme de MPI. LAM est un environnement d'exécution basé sur le démon de « runtime ». LAM-MPI combine cet environnement avec une bibliothèque qui met en place les APIs de MPI. Il est l'un des environnements de calcul les plus populaires de MPI dans le monde scientifique.

Nous mettons en œuvre l'algorithme GMRES(m) sur ce système de grille XtremWeb ainsi que dans l'environnement de LAM-MPI. Nous ouvrons une voie neuve et significative pour résoudre les problèmes décrits par des systèmes linéaires à grande échelle, Selon les résultats que nous avons obtenus lors de tests nombreux, nous remarquons surtout les avantages suivants du système XtremWeb :

Nous pouvons exécuter nos applications sur des sites hétérogènes qui sont très fréquents sur Internet grâce à l'exécution des applications natives dans un contexte sécurisé ainsi que des applications java.

Nous pouvons déployer facilement ce système XtremWeb en mettant à jour automatiquement le code par un système d'administration des ressources à distance.

Nous pouvons sécuriser toutes les connexions en appliquant les politiques d'authentification par défi cryptographique, d'exécution de code natif au site local sécurisé, de contrôle administratif des utilisateurs et des postes de travail.

XtremWeb a également une bonne tolérance aux pannes. Un composant logiciel du système peut être interrompu sans produire de faute sur les autres composants du système et redémarré en récupérant son état précédant la faute. C'est certainement un avantage

remarquable en environnement d'Internet comprenant de très nombreux dispositifs volatiles.

Pourtant nos tests montrent également quelques inconvénients de XtremWeb. La centralisation de gestion des tâches par un Coordinateur devient un embouteillage de ce système. Heureusement, une version décentralisée d'XtremWeb baptisée XtremWeb-CH[♣] est développée par EIG (Ecole d'Ingénieurs de Genève) en Suisse. Il se compose d'un module de programmation, au dessus de XtremWeb, qui insère des tâches, en marche, par une description d'XML. Il lance les tâches sans dépendances avant, récupère les résultats et puis met à jour les tables de la base de données (DB). Dans cette version, les Travailleurs peuvent se communiquer directement entre eux, seules des références de données sont envoyées au Coordinateur [50][51].

Nous profitons des avantages présentés pour réaliser notre méthode GMRES(*m*). Mais à cause des communications intensives liées à la nature de cet algorithme, le coût des communications est très élevé même quand les dimensions des matrices testées sont petites.

Après avoir réussi l'implantation de la méthode hybride GMRES(*m*)/LS-Arnoldi dans le cas réel, nous tentons de trouver une résolution dans le cas complexe. Les résultats mettent en évidence la nécessité d'approfondir la théorie mathématique concernant ce cas. Dans l'avenir, nous aurons donc à développer avec rigueur la théorie mathématique de cette méthode hybride dans le cas complexe. Nous espérons ainsi obtenir une grande amélioration pour la résolution des grands systèmes linéaires à variables complexes.

La croissance exponentielle de nombre des dispositifs (Ordinateurs, PDAs, Téléphone Mobile etc.) connectés sur Internet, le développement rapide des technologies de réseau haut débit, et les déploiements des technologies de télécommunication de la troisième génération, WiFi, WiMax pour les dispositifs personnels mobiles très nombreux, puis la disponibilité de plus en plus de ressources informatiques fréquemment inexploitées offrent une opportunité d'assembler un supercalculateur virtuel. Il pourra globaliser l'accès aux ressources et aux données. Cette ambition est l'objectif principal des

[♣] http://www.xtremwebch.net

systèmes de grilles de calcul. Et la plate-forme d'expérimentation XtremWeb est actuellement l'un des systèmes de grille léger les plus réussis.

La portabilité de notre implantation sur le système XtremWeb est confirmée par nos tests. A l'avenir, en utilisant ce système générique de grille XtremWeb, nous profiterons d'une très grande quantité de ressources reliées par Internet avec davantage de Travailleurs engagés à travers des réseaux de haut débit, de plus en plus rapides à résoudre sans difficulté les systèmes linéaires à très grande échelle jusqu'à une dimension de plusieurs millions.

Nous avons également montré dans cette thèse trois types de machines aptes à faire du calcul intensif : supercalculateur, PCs, grappe, et les environnements de calcul parallèle ou distribué associés : IBM RS6000/SP, XtremWeb, LAM-MPI. Pour résoudre un problème précis, comment choisir l'implantation la plus appropriée est toujours une question intéressante pour notre recherche.

Le champ de recherche ouvert par le calcul de grille est large, et renouvelle en partie les problématiques associées à la conception des grilles par la quantité des ressources mises en jeu et par la volatilité de ces ressources. Les études que nous avons présentées sont une contribution à celles qu'il reste à mener. Grâce à la conception de grille, à l'avenir nous espérons qu'une interface homme-machine uniforme sera réalisée pour intégrer et utiliser ces ressources différentes quels que soient les supercalculateurs, les grappes, les PCs et aussi un système de grille universel qui pourra intégrer et faire collaborer tous les systèmes de calcul différents.

Bibliographique

[1]Y. SAAD, M. H. Schultz, GMRES: a generalized GMRES algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Statist. Compt., 14(1993) 461-469

[2]M.J.FLYNN, Very high-speed computing systems, Proceedings of the IEEE, (1996) 1901-1999

[3]M. CONSNARD, D. TRYSTRAM, Algorithmes et architechectures parallèles, InterEditions, Paris, 1993

[4]J. CULLUM, A. GREENBAUM, Relations between Galerkin and norm-minimizing iterative methods for solving linear systems, SIAM J. Matrix Anal. Appl., 17(1996) 223-247

[5]A.BEGUELIN, J. DONGARRA, A. GEIST, R. MANCHEK, V. SUNDERAM, A user's guide to PVM-Parallel Virtual Machine, Research Report ORNL/TM11826, Oak Ridge National Laboratory, 1992

[6]Message Passing Interface Forum, Document for a standard Message Passing Interface, 1993

[7]Une brève introduction de IBM RS6000/SP3. www.univ-lille1.fr/calcul-intensif/corinfo.htm

[8]Y.SAAD. SPARSKIT: a basic tool kit for sparse matrix computations. RIACS, NASA Ames Research Center, 1991.

[9]S. PETITON. Data parallel sparse matrix computation on CM2 and CM5 for iterative methods. University of Minnesota, AHPCRC, 1993

[10] S.PETITON. Contribution à une méthodologie globale pour le calcul scientifique parallèle. Thèse d'habilitation, Université de Paris VI, 1993

[11] H. F. WALKER. Implementation of the GMRES method using Householder transformation. SIAM. J. Sci. Statist. Comput., 9:152, 1988

[12] J. ERHEL. A parallel GMRES version for general sparse matrices. Electronic Transactions on Numerical Analysis, 3:160-175, 1995

[13] Y. SAAD. Etude de la convergence du procédé d'Arnoldi pour le calcul d'éléments propres de grande matrices non symétriques. 1979

[14] Y. SAAD. Numerical methods for large eigenvalues problems. Manchester University Press Series in Algorithms and Architectures for Advanced Scientific Computing, 1993

[15] Y. SAAD. Least squares polynomials in the complex plane and their use for solving nonsymmetric linear systems. SIAM J. Sci. Statist. Comput., 7:155-169, 1987.

[16] SEBASTIEN VEIGNEAU. PVM un outil de distribution de calculs sur une machine virtuelle. Institut Gaspard Monge, Université de Marne-la-Vallée

[17] V.S.SUNDERAM, PVM: A Framework for Parallel Distributed Computing, Concurrency: Practice and Experience, Vol 2, N° 4, pp 315-339, Dec 1990.

[18] AL.GEIST and V.S. SUNDERAM, Network Based Concurrent Computing on the PVM System,Concurrency: Practice and Experience, Vol 4, N° 4, pp 293-311, June 1992.

[19] A.ESSAI, G. BERGERE, and S. PETITON, Heterogeneous parallel hyrid GMRES/LS-Arnoldi. Ninth SIAM Conference on Parallel Processing for Scientifc Computing, 1999

[20] Un bref introduction de IBM RS6000/SP4 de CINES. http://www.cines.fr/materiels0.html

[21] Foster, I. and Kesselman, C. (eds) (1999) *The Grid: Blueprint for a New Computing Infrastructure*. San Francisco, CA: Morgan Kaufmann.

[22] The Global Grid Forum Web Site, http://www.gridforum.org.

[23] The Globus Project Web Site, http://www.globus.org.

[24] Berman, F., Fox, G. and Hey, T. (2003) *Grid Computing: Making the Global Infrastructure a Reality*. Chichester: John Wiley & Sons.

[25] Web Site associated with book, *Grid Computing: Making the Global Infrastructure a Reality*, http://www.grid2002.org.

[26] Fran Berman, Geoffery Fox, and Tony Hey, The Grid: past, present, future (2003) Grid Computing—Making the Global Infrastructure a Reality. ISBN: 0-470-85319-0

[27] Franck Cappello, Abderrahmane Djilali, Gilles Fedak, Cécile Germain, Oleg Lodygensky, Vincent Néri, , XtremWeb : une plate-forme de recherche sur le Calcul Global et Pair à Pair - Calcul réparti à grande échelle Metacomputing ch 6, Françoise Baude ,Hermes Sciences - Lavoisier, 2002

[28] Gille Fedak XtremWeb: une plate-forme pour l'étude expérimentale du calcul global pair à pair. Thèse, Université de Paris XI, 2003

[29] Greg Burns, Raja Daoud, and James Vaigl, "LAM: An open cluster environment for MPI," In John W. Ross, editor,Proceedings of Supercomputing Symposium '94, pages 379-386. University of Toronto, 1994

[30] G. D. Burns, “The Local Area Multicomputer,” in Proceedings of the Fourth Conference on Hypercube Concurrent Computers and Applications, ACM Press, March 1989

[31] A.Essai. Méthode hybride parallèle hétérogène et méthodes pondérées pour la résolution des systèmes linéaires Thèse de l’Université de Lille I, ANO, 1999

[32] J.J. PESQUE, Méthodes itératives sur quelques familles de matrices creuses de la variable complexe. Rapport CEA/CESTA, DO 353, Août 2002.

[33] Gilles Fedak, Cécile Germain, Vincent Néri and Franck Cappello. XtremWeb : A Generic Global Computing System. 0-7695-1010-8/10, 2001 IEEE

[34] http://www.globus.org.

[35] Frank Cappello, Samir Djilai, Gilles Fedak, el. Computing on large-scale distributed systems: XtremWeb architecture, programming models, security, tests and convergence with grid. Future Generation Computer System 21(2005) 417-437

[36] http://setiathome.ssl.berkeley.edu

[37] O.Lodygensky, G. Fedak, F. Cappello, V. Neri, M.Livny and D. Thain. XtremWeb & Condor: sharing resources between Internet connected Condor pools.GP2PC2003, Global and Peer-to-Peer Computing on Large Scale Distributed Systems. Tokyo, Japan, May.2003.

[38] Al Geist, William Gropp, Steve Huss-Lederman, Andrew Lumsdaine, Ewing Lusk, William Saphir,Tony Skjellum, and Marc Snir. MPI-2: Extending the Message-Passing Interface. In Luc Bouge, PierreFraigniaud, Anne Mignotte, and Yves Robert, editors, Euro-Par ’96 Parallel Processing, number 1123 in Lecture Notes in Computer Science, pages 128–135. Springer Verlag, 1996.

[39] Message Passing Interface Forum. MPI: A Message Passing Interface. In Proc. of Supercomputing'93, pages 878–883. IEEE Computer Society Press, November 1993.

[40] http://www.mpi-forum.org

[41] http://www.lam-mpi.org

[42] Jeffrey M. Squyres, Brian Barrett, and Andrew Lumsdaine. MPI collective operations system services interface (SSI) modules for LAM/MPI. Technical Report TR577, Indiana University, Computer Science Department, 2003.

[43] Jeffrey M. Squyres, Brian Barrett, and Andrew Lumsdaine. Request progression interface (RPI) system services interface (SSI) modules for LAM/MPI. Technical Report TR579, Indiana University, Computer Science Department, 2003.

[44] Jeffrey M. Squyres, Brian Barrett, and Andrew Lumsdaine. The system services interface (SSI) to LAM/MPI. Technical Report TR575, Indiana University, Computer Science Department, 2003.

[45] The LAM/MPI Team. LAM/MPI Installation Guide. Open Systems Laboratory, Pervasive Technology, Labs, Indiana University, Bloomington, IN, 7.0 edition, May 2003.

[46] www.xtremweb.net, XtremWeb User Guide, Installation, Usage and Programming, Edition 1.3.0, for XtremWeb version 1.3.0

[47] Sato.M; Boku.T; Takahashi.D ; OmniRPC: a grid RPC system for parallel programming in cluster and grid environment.. Cluster Computing and the Grid, 2003. Proceedings. CCGrid 2003. 3rd IEEE/ACM International Symposium on 12-15 May 2003 Page(s):206 - 213

[48] Guy BERGERE, Contribution à une programmation parallèle hétérogène de méthodes numériques hybrides, thèse 1999.9. USTL

[49] http://www.llnl.gov/computing/tutorials/ibm_sp/#Switch

[50] Lamine Aouad, Serge Petiton. Experimentations and Programming Paradigms for Matrix Computing on Peer to Peer Grid. Proceeding of the 5th IEEE/ACM International Workshop on Grid Computing (GRID'04) 1550-5510/04 IEEE

[51] XtremWeb-CH : Un outil Peer-To-Peer orienté Calcul Intensif. N. Abdennadher. "Journée Scientifique des Technologies de l'Information et de la communication de la HES-SO". Yverdon, Switzerland, 12 May 2004

[52] A.Geist,A.Beguelin,J.Dongarra,W.Jiang,R.Manchek and V.Sunderam. PVM Parallel Virtual Machine:A Users Guide Tutorial for Networked ParallelComputing The MIT Press,1994

[53] Z.Hafidi . MARS : Un environnement de programmation parallèle adaptative dans les réseaux de machines hétérogènes multi-utilisateurs. Thèse de l'Université de Lille I, LIFL,1998

[54] Z.Hafidi,E.-G.Talbi, and J-M.Geib . MARS : Un ordonnanceur adaptatif d'applications parallèles dans un environnement multi-utilisateurs. Université de Lille I, LIFL,1998

[55] A.T.Ogielski and W.Aiello. Sparse matrix computations on parallel processor arrays. SIAM J.Sci.Comput., 14(3) : 519-530,1993

[56] J.Saltz, S.Petiton, H. Berryman, and A.Rifkin. Performance effects of irregular communication patterns on massively parallel mutltiprocessors. Journal of Parallel and Distributed Computing, 13(2):202, 191

[57] J.P.Malmquist and E.L.Robertson. On the complexity of partitioning sparse matrix representation. BIT, 24:60-68, 1984

Table des figures

TABLE DES FIGURES

TABLE DES FIGURES

Liste des tableaux